



UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

iteePhD
information technology
electrical engineering



DIPARTIMENTO DI ECCELLENZA 2018
DI ECCELLENZA 2022
DIETI
DIPARTIMENTO
DI ECCELLENZA
2023 - 2027

Università degli Studi di Napoli Federico II
Ph.D. Program in
Information Technology and Electrical Engineering
XXXVI Cycle

THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Deep Learning for Railway Safety and Maintenance: Methodologies and Applications

by

LORENZO DE DONATO

Advisor: Prof. Valeria Vittorini

Co-advisors: Prof. Carlo Sansone and Prof. Francesco Flammini



SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

*To my family,
an invaluable source of strength and inspiration.*

DEEP LEARNING FOR RAILWAY SAFETY AND MAINTENANCE: METHODOLOGIES AND APPLICATIONS

Ph.D. Thesis presented
for the fulfillment of the Degree of Doctor of Philosophy
in Information Technology and Electrical Engineering
by

LORENZO DE DONATO

October 2023



Approved as to style and content by

Valeria Vittorini

Prof. Valeria Vittorini, Advisor

Valeria Vittorini

Prof. Carlo Sansone, Co-advisor

Carlo Sansone

Prof. Francesco Flammini, Co-advisor

Università degli Studi di Napoli Federico II

Ph.D. Program in Information Technology and Electrical Engineering

XXXVI cycle - Chairman: Prof. Stefano Russo



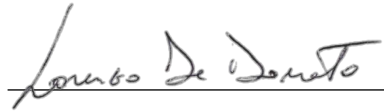
<http://itee.dieti.unina.it>

Candidate's declaration

I hereby declare that this thesis submitted to obtain the academic degree of Philosophiæ Doctor (Ph.D.) in Information Technology and Electrical Engineering is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.

Parts of this dissertation have been published in international journals and/or conference articles (see list of the author's publications at the end of the thesis).

Napoli, December 13, 2023

A handwritten signature in dark ink, reading "Lorenzo De Donato", is written over a horizontal line.

Lorenzo De Donato

Abstract

This Thesis explores the adoption of Deep Learning (DL) for railway safety and maintenance applications, concentrating on solutions leveraging video and audio analysis. After a systematisation of relevant Artificial Intelligence (AI) and railway concepts, the Thesis evaluates the opportunities offered by DL in Autonomous Trains (ATs) environment perception and non-intrusive inspections of safety-critical railway assets for maintenance purposes. Then, the Thesis focuses on the main challenges concerning the integration of AI into safety-related railway applications, evaluating certifiability and technical issues.

The Thesis first analyses both aspects methodologically. The concept of “Grade of Intelligence (GoI)” is introduced for the delineation of high-level guidelines that could support the step-wise integration of AI and DL in the development of *Intelligent Train Control (ITC)* systems. Then, a seven-step methodology is presented for the development of DL solutions.

Hence, the Thesis proceeds with the discussion of two experimental case studies exemplifying the adoption of the aforementioned methodology. The first tackles the problem of anomaly detection, i.e., the identification of unknown objects, on rail tracks. The second deals with the continuous monitoring of Level Crossings (LCs) for predictive maintenance. These were chosen for their relevance to safety within the context of study, enabling an exploration of techniques and tools to address – or exploit – specific constraints within the application domain and the lack of suitable datasets for DL models development.

Keywords: Artificial Intelligence, Artificial Vision, Railway Transport, Autonomous Train Driving, Smart Maintenance, Level Crossing.

Sintesi in lingua italiana

La presente Tesi esplora l'utilizzo del Deep Learning (DL) per la sicurezza (safety) e la manutenzione ferroviaria, con particolare riferimento a soluzioni basate sull'analisi video e audio. Dopo una sistematizzazione dei concetti rilevanti inerenti all'Intelligenza Artificiale (IA) e al settore ferroviario, la Tesi valuta le opportunità offerte dal DL nella percezione dell'ambiente nel contesto dei treni a guida autonoma e nelle ispezioni non invasive di sistemi ferroviari critici a scopi manutentivi. Inoltre, la Tesi si concentra sulle principali problematiche riguardanti l'integrazione dell'IA nelle applicazioni ferroviarie legate alla sicurezza, valutandone gli aspetti di certificabilità e i problemi tecnici.

In primo luogo, la Tesi analizza entrambi gli scenari da un punto di vista metodologico. Viene introdotto il concetto di “Grado di Intelligenza” per definire delle linee guida atte a supportare l'integrazione dell'IA e del DL nello sviluppo dei sistemi di Controllo Ferroviario Intelligente. Successivamente, viene presentata una metodologia in sette fasi per lo sviluppo di soluzioni basate sul DL.

La tesi prosegue con la discussione di due casi di studio che illustrano l'adozione della metodologia sopra menzionata. Il primo affronta il problema della rilevazione di anomalie sulle rotaie, cioè l'identificazione di oggetti sconosciuti. Il secondo tratta il monitoraggio continuo dei passaggi a livello per la manutenzione predittiva. Questi sono stati scelti per la loro rilevanza per la sicurezza nel contesto di riferimento e hanno consentito di investigare tecniche per affrontare – e ove possibile sfruttare - i particolari vincoli posti dal dominio applicativo, e la mancanza di opportuni dataset per lo sviluppo dei modelli di DL.

Parole chiave: Intelligenza Artificiale, Visione Artificiale, Trasporto Ferroviario, Treni a Guida Autonoma, Manutenzione Intelligente, Passaggio a Livello.

Acknowledgements

The author's work has been supported by a PhD scholarship funded by "CINI - Consorzio Interuniversitario Nazionale per l'Informatica", partially on the Horizon 2020 Shift2Rail project "RAILS - Roadmaps for AI integration in the rail Sector".

The RAILS project received funding from the Shift2Rail Joint Undertaking (JU) under the European Union's Horizon 2020 research and innovation programme under grant agreement n. 881782. The JU receives support from the European Union's Horizon 2020 research and innovation program and the Shift2Rail JU members other than the Union.



Disclaimer

The information and views set out in this document are those of the author and do not necessarily reflect the official opinion of Shift2Rail JU. The JU does not guarantee the accuracy of the data included in this document. Neither the JU nor any person acting on the JU's behalf may be held responsible for the use which may be made of the information contained therein.

Contents

Abstract	i
Sintesi in lingua italiana	iii
Acknowledgements	v
List of Acronyms	xi
List of Figures	xviii
List of Tables	xx
1 Introduction	1
1.1 Research Context and Motivation	1
1.1.1 The RAILS project	2
1.1.2 AI for Rail Safety and Automation	3
1.1.3 AI for Railway Maintenance	4
1.1.4 Challenges for AI	5
1.2 Focus and Contribution	5
1.3 Structure	7
2 Context and Background	9
2.1 A Taxonomy for AI in Railways	10
2.1.1 Reference Taxonomy	11
2.1.2 From Machine Learning to Deep Learning	14

2.1.3	Deep Learning for Computer Vision and Audio Processing	18
2.2	Overview of Relevant Railway Concepts and Systems	21
2.2.1	Automatic Train Driving Systems	21
2.2.2	Maintenance and Audio-Video Analytics	23
2.3	A Review of DL Solutions Enhancing Railway Safety and Maintenance Applications	25
2.3.1	Obstacle Detection on Rail Tracks	25
2.3.2	Assets Monitoring through Video/Audio Data	35
2.4	Challenges for DL in Railways	38
2.4.1	Certiability of AI Systems	38
2.4.2	Data Issues: Quality and Availability	45
3	Introducing DL in Intelligent Train Control	51
3.1	An Example Architecture for Autonomous Trains	52
3.2	Conceptual Shift: Grades of Intelligence	53
3.3	Structural Needs: Implementing Grades of Intelligence	56
3.4	From Limited/No Autonomy to Full Autonomy	62
4	A Reference Methodology for Developing DL Applications	65
4.1	Focus and Boundaries	65
4.2	A 7-Step Methodology	67
4.2.1	Definition of the Problem Statement	67
4.2.2	Identification of the Approach and Selection of DNNs	69
4.2.3	Identification and/or Generation of Suitable Datasets	70
4.2.4	DNNs Training, Validation, Test, and in-depth Experiments	73
4.2.5	Discussion of Results	75
4.2.6	Structured Evaluation of the Approach	75
4.2.7	Identification of Future Directions and Recommendations	78

5	Case Study 1: Vision-based Obstacle Detection on Rail Tracks	79
5.1	Problem Statement	79
5.2	Identification of the Approach and Selection of DNNs	82
5.2.1	A Modular Approach	82
5.2.2	Selection of DNNs	84
5.3	Identification and/or Generation of Suitable Datasets	86
5.3.1	Domain-Specific Characteristic: Constrained Paths .	86
5.3.2	Virtual Railway Scenarios for Synthetic Data Generation	88
5.3.3	Generation of Suitable Datasets	90
5.4	DNNs Training, Validation, Test, and in-depth Experiments	97
5.4.1	Rail Tracks Identification: U-Net	97
5.4.2	Anomaly Detection: SSIM-VQ-VAE	105
5.5	Discussion of Results	113
5.5.1	Analysing Results and Future Implementations . . .	114
5.5.2	Answering RQs and Evaluating KPIs	117
5.6	Structured Evaluation of the Approach	119
5.6.1	Strengths (↑) and Weaknesses (↓)	120
5.6.2	Opportunities (↑) and Threats (↓)	122
5.7	Identification of Future Directions and Recommendations .	124
6	Case Study 2: Level Crossings Monitoring through Audio/Video Data	127
6.1	Problem Statement	127
6.2	Identification of the Approach and Selection of DNNs	130
6.2.1	A Modular Approach	131
6.2.2	Selection of DNNs	133
6.3	Identification and/or Generation of Suitable Datasets	135
6.3.1	Warning Bell Dataset Generation	136

6.3.2	Protective Barrier Dataset Generation	140
6.4	DNNs Training, Validation, Test, and in-depth Experiments	144
6.4.1	Warning Bell Analysis Module: A VGGish-inspired CNN	144
6.4.2	Protective Barrier Analysis Module: YOLOv5	161
6.5	Discussion of Results	175
6.5.1	Analysing Results and Future Implementations . . .	175
6.5.2	Answering RQs and Evaluating KPIs	182
6.6	Structured Evaluation of the Approach	184
6.6.1	Strengths ($\uparrow$) and Weaknesses ($\downarrow$)	185
6.6.2	Opportunities ($\uparrow$) and Threats ($\downarrow$)	187
6.7	Identification of Future Directions and Recommendations .	189
7	Conclusions	193
	Bibliography	197
	Author's publications	213

List of Acronyms

The following acronyms are used throughout the Thesis.

AI	Artificial Intelligence
AoE	ATO over ETCS
AT	Autonomous Train
ATC	Automatic Train Control
ATO	Automatic Train Operation
ATP	Automatic Train Protection
ATS	Automatic Train Supervision
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
DT	Digital Twin
ETCS	European Train Control System

GAN	Generative Adversarial Network
GoA	Grade of Automation
GoI	Grade of Intelligence
GTA	Grand Theft Auto
I2X	Infrastructure to Everything
ITC	Intelligent Train Control
ITO	Intelligent Train Operation
ITP	Intelligent Train Protection
IoT	Internet of Things
IoU	Intersection over Union
KPI	Key Performance Indicator
LC	Level Crossing
LoI	Level of Intelligence
mAP	mean Average Precision
ML	Machine Learning
OD	Obstacle Detection
PoC	Proof-of-Concept
RQ	Research Question
SIL	Safety Integrity Level
SNR	Signal-to-Noise Ratio

SS	Semantic Segmentation
SSIM	Structural Similarity Index Measure
SWOT	Strenghts, Weaknesses, Opportunities, and Threats
T2X	Train to Everything
TL	Transfer Learning
TTS	Frame-Level Trimmed Training Set
TJS	Frame-Level Trimmed Joint Set
VQ-VAE	Vector Quantised-Variational Autoencoder

List of Figures

2.1	A Class Diagram for AI , excerpted from [111]	12
2.2	A Class Diagram for ML , excerpted from [111]	14
2.3	Examples of Transfer Learning	17
2.4	A High-Level Classification of Obstacle Detection Systems .	27
2.5	A Classification of Approaches for Vision-based On-train OD	28
2.6	Examples of Rail Tracks Identification Approaches	29
2.7	Overview of Some Railway Relevant Assets and Related Components [24]	35
2.8	High-Level Picture of Approaches for Railway Assets Defect Detection through Video/Audio Data	36
2.9	Risk-based Classification of AI Systems [90]	40
2.10	Exemplification of Causes that could Fool a DNN	42
2.11	Introducing AI in Autonomous Trains [44]	43
3.1	An Example Architecture for Autonomous Trains	52
3.2	Grade of Intelligences	55
3.3	An Example Architecture Towards GoI4	58
4.1	A 7-Step Methodology	66
5.1	An Architecture for Vision-Based OD Systems	83

5.2	Building a DNN for each Railway Line	87
5.3	Which Mask should be Predicted?	87
5.4	Building a DNN for each Path	88
5.5	Scenario Generated in RoadRunner with Sample Frames . .	89
5.6	Overview of the Scenario Generated in Unreal Engine 5 . .	90
5.7	Semi-Automatic Labelling based on Self-Training and TL .	91
5.8	Evaluation of Generated Pseudo-labels	93
5.9	Examples of Transformations on Frame from RoadRunner FreeTrack Video	94
5.10	Paths Simulated in Unreal Engine 5 and Related Sample Frames	96
5.11	U-Net Pre-training on Modified RailSem19: Training Curves	98
5.12	U-Net Training Curves on RR Dataset	99
5.13	U-Net Predictions on RR Dataset Test Set Frames	100
5.14	U-Net Correct Prediction of Rail Tracks Ahead the Train: MainTrack and SideTrack Datasets	101
5.15	U-Net Noisy Prediction of Frames with Obstacle	102
5.16	U-Net Pipeline: Prediction and Post-Processing	104
5.17	Insights on VQ-VAE [144]	107
5.18	Training SSIM-VQ-VAE on <i>NoAnomaly</i> and <i>Augmented-</i> <i>NoAnomaly</i> : Losses	108
5.19	Training SSIM-VQ-VAE on <i>NoAnomaly</i> and <i>Augmented-</i> <i>NoAnomaly</i> : SSIM Scores	109
5.20	SSIM-VQ-VAE Reconstructions on <i>NoAnomaly</i> and <i>Aug-</i> <i>mentedNoAnomaly</i> Test Data	110
5.21	Performances of SSIM-VQ-VAE trained on <i>NoAnomaly</i> (or- ange) and on <i>AugmentedNoAnomaly</i> (blue) evaluated on the <i>AugmentedNoAnomaly</i> Test Set	110
5.22	Examples of Anomaly Maps Generated through SSIM-VQ-VAE and related Post Processing (Heatmaps)	111

5.23	SSIM Scores Generated by SSIM-VQ-VAE on <i>WithCar</i> Frames	112
5.24	SSIM-VQ-VAE False Positive given Tree's Shadow	113
5.25	Rail Track Segmentation and Anomaly Detection Pipeline .	114
5.26	A SWOT Analysis for the <i>Rail Track segmentation and Anomaly Detection Pipeline</i>	120
6.1	An Architecture for Level Crossing Intelligent Monitoring (Image Extracted from Grand Theft Auto V (<i>GTA V</i>)) . . .	132
6.2	Protective Barrier Motion Extraction Pipeline	135
6.3	Distribution of Collected Audio Clips	137
6.4	Audio Data Pre-Processing	138
6.5	Example of Framing and Trimming Operations	139
6.6	Example Frames extracted from <i>GTA</i> Videos	140
6.7	Examples of Patches Obtained from <i>GTA</i> Video Frames . .	141
6.8	Example of Augmented (DayClear) Patches	143
6.9	VGGish High-Level Architecture	144
6.10	Examples of stable and unstable configurations	147
6.11	Training Curves of the Best Performing Configuration . . .	149
6.12	Cross-Validation Folds and Partitions	150
6.13	Cross-Validation Results	151
6.14	Training Performances of the Best Configuration on <i>Parti-tion 1</i> under Different Weights Compositions.	152
6.15	Frame-Level and Audio-Level Accuracy on Test Sets per Reduced <i>TJS</i>	156
6.16	YOLOv5s Performances on Validation Set in case of Stochastic Gradient Descent and Adam Optimizer	164
6.17	Contribution of <i>TL</i> and Augmentations in YOLOv5s	166
6.18	Barrier Movement Extracted from DayClear Video Clips . .	169
6.19	Barrier Movement Extracted from NightClear Video Clips .	169
6.20	Undetected Barriers from DayClear Clips	170

6.21	Undetected Barriers from NightClear Clips	170
6.22	Prediction Confidences for DayClear Video Clips	171
6.23	Prediction Confidences for NightClear Video Clips	171
6.24	Patches from NighClear Clips with Low Confidence Values. C: Confidence Value - H: Detected Height.	173
6.25	Simulated Barrier's Malfunctions	180
6.26	A SWOT Analysis for the <i>LC Intelligent Monitoring System</i>	185

List of Tables

2.1	Automatic Train Systems, excerpted from [44]	21
2.2	<i>Ethic Principles</i> and <i>Key Requirements</i> from [38]	39
5.1	Data Augmentation Transformations (Transf.)	94
5.2	Distribution of Samples in <i>RR Dataset</i>	95
5.3	Distribution of Samples in <i>MainTrack</i> and <i>SideTrack</i> Datasets	96
5.4	Distribution of Samples in <i>NonAnomaly</i> and <i>Augmented-NoAnomaly</i> Datasets	106
5.5	Performances of SSIM-VQ-VAE at the Best Epoch(s)	109
6.1	Samples Distribution in Audio-Level and Frame-Level Sets	139
6.2	Data Augmentation Transformations (Transf.)	142
6.3	Mapping GTA <i>Video Types</i> and Transformations	142
6.4	Configurations Hyperparameters and Parameters	145
6.5	Best Configurations after Hyperparameters Tuning	148
6.6	Best Configurations after Parameters Tuning	149
6.7	Confusion Matrices at the Frame-Level and the Audio-Level	154
6.8	Precision, Recall, and F1-score per Class	154
6.9	Classification Performance on Warning Bell Frames in Test Set when Trained on “Italian only TJS ”	157

6.10	Classification Performance on Warning Bell Frames in Test Set when Trained on “British only TJS ”	157
6.11	Classification Performance on German and French <i>warning bells</i> when Trained on TJS	158
6.12	Classification Performances on Noisy Test Sets	160
6.13	YOLOv5s Performances at different Fitness Functions	165
6.14	YOLOv5s Performances on Validation Set with/without TL	167
6.15	YOLOv5s Performances on Training, Validation, and Test Sets	167
6.16	Average (Avg.) Confidence (Conf.) Values per Barrier Position	172
6.17	Pipeline’s Computation Times in milliseconds (ms)	174
6.18	Metrics for Anomaly Evaluation in Barrier Movement	181

Introduction

Artificial Intelligence, in all its facets, is largely catching the attention of researchers and practitioners worldwide for the opportunities it introduces in empowering current practices or establishing new processes in different domains. *Rail transport*, despite some complications mainly linked to the safety-critical nature of most of the railway systems, is one of those that have recently seen exponential growth in the investigation of [AI](#) applications [[138](#), [24](#), [113](#)] which, together with other prominent technologies like blockchain, cloud computing, and Internet of Things ([IoT](#)), is seen as one of the main enablers for future railways [[139](#)].

1.1 Research Context and Motivation

The rail sector has always played a central role in people and goods mobility. At the European level, it is considered the core of public transport as well as an essential contributor to states' economy; its pivotal role is destined to grow even further in the coming years, where railways will visionary become, by 2050, the backbone of future multi-modal mobility interconnecting ground, air, and water transport [[94](#), [139](#)]. This makes the railways an ideal stage to showcase the potential that [AI](#) has in concretely benefiting future society, both in terms of services and the economy. The key research pillars towards the concretisation of the vision above include, but are not limited to: i) advancements in *railway automation*, by implementing mechanisms oriented at, among others, increasing railway capacity

while making trains capable of taking driving decisions in autonomy (i.e., Autonomous Trains (ATs)) and running closer to each other (i.e., Virtual Coupling); ii) *intelligent life-cycle management* of railway assets, including procedures supporting the shift towards their digitalised, on-line monitoring, and intelligent prognosis and health assessment/forecasting; and iii) *sustainability*, concerning the research towards greener propulsion systems and energy consumption optimisation [94, 139].

The research conducted under the *Shift2Rail*¹ (before) and *Europe's Rail*² (now) banners are largely and tangibly contributing to the railway technological advancement, investigating the applicability of cutting-edge technologies (including AI) for the development of next-generation railway systems.

1.1.1 The RAILS project

Among the plethora of projects conducted under the two banners just mentioned, the Horizon 2020 Shift2Rail project “RAILS - Roadmaps for AI integration in the rail sector”³ had a key role in the definition of the PhD research activities of this Thesis' author.

The project aimed at investigating the potential of AI approaches in the rail sector and contributing to the definition of roadmaps for or future research in the areas of *train safety and automation*, *assets maintenance and inspection*, and *traffic planning and management*. Hence, RAILS encompassed a number of theoretical and experimental activities including the development of the following Proofs-of-Concept (PoCs): i) “Vision-based Obstacle Detection on Rail Tracks” and “Cooperative Driving for Virtual Coupling of Autonomous Trains” in the context of railway safety and automation; ii) “Smart Maintenance at LCs” and “AI-based Rolling Stock Rostering” for what concerns maintenance and inspection; and iii) “Primary Delay Prediction” and “Incident Attribution Analysis” in relation to traffic planning and management. The results obtained show that AI would be a disruptive catalyst to innovate, support, and/or add functionalities to “traditional” railway systems and practices, i.e., those adopted in railways (even for decades) which do not involve any “intelligent behaviour”.

¹<https://rail-research.europa.eu/about-shift2rail>

²<https://rail-research.europa.eu/about-europes-rail>

³<https://rails-project.eu>

This Thesis reports and/or extends some of the results obtained in the context of the RAILS project, including methodologies and experimental activities falling within the areas of *train safety and automation* and *assets maintenance and inspection*. Particular focus is posed on: i) the definition of a *taxonomy of AI in railways*; ii) the *identification of Grades-of-Intelligence (GoIs)* and a *vision towards Intelligent Train Control*; and iii) the PoCs of “*Vision-based Obstacle Detection on Rail Tracks*” and “*Smart Maintenance at Level Crossings*” (from both methodological and experimental perspectives). These have been among the primary research topics of this Thesis’ author during his PhD studies and to which he has largely contributed as the main (and only) or one of the primary researchers within the RAILS project.

1.1.2 AI for Rail Safety and Automation

In line with the vision supported by the European Rail Research Advisory Council [94, 139], a huge effort is being put into the automation of railway lines, which can be conceptually translated into finding mechanisms to make trains capable of running unattended, precisely respecting the timetable, and saving as much energy as possible, while ensuring a high level of safety. From this perspective, railway lines can be subdivided into two macro categories [95, 44]: *segregated*, i.e., railway lines that are completely isolated from external threats by means of physical barriers (e.g., metro lines with platform screen doors), and *open*, which can be defined by contrast as those railway lines that present intersection points, like Level Crossings and stations, from where external agents could access rail tracks. Up to date, *unattended operation* – which, in the sector, is indicated by a Grade of Automation (GoA) [67] Level 4 (GoA4) – has already been achieved in some *segregated* railways without the necessity of introducing particular AI functionalities [44]. External threats are almost nonexistent and all other kinds of anomalies that could affect the safety of running trains are handled by dedicated systems (e.g., the Automatic Train Protection (ATP), or equivalent) which continuously monitor trains’ automatic operations and apply protection mechanisms when required [44]. Differently, the research towards driverless trains in *open* environment is still ongoing [44]. In this context, one of the main challenges comes from the fact that, while some hazardous events could be managed through formalisms or rule-based mechanisms (referred to as *rule-based safety* in

[124]), there are some phenomena which can be handled only by human operators (referred to as *managed safety* in [124]). Here is where AI is showing to be capable of playing a very central role, providing trains with the adequate “skills” to detect or predict anomalies both in onboard components and within the surrounding environment. A striking example is given by obstacles on rail tracks, a topic tackled in this Thesis, which would be of primary importance for the safety of *ATs in open environments*.

1.1.3 AI for Railway Maintenance

Similar considerations apply to maintenance and inspection activities, which, although not directly, have a huge impact on railway safety [89]. In this context, AI has largely and tangibly proven to be effective, supporting the paradigm shift from scheduled and corrective to continuous and predictive maintenance. Indeed, current on-field inspection activities are traditionally carried out by human operators following fixed scheduling (e.g., weekly or monthly), when traffic is suspended, and, in some cases, supported by costly equipment [69, 24, 151]. In addition to being costly and (potentially) conducted in conditions that are unfavourable for the proper detection of anomalies (e.g. at night, when visibility is not optimal), these practices may also not allow for the timely detection of malfunctions occurring between subsequent inspections. Approaches based on AI can help to mitigate all these factors, operating continuously to promptly detect anomalies and predict possible future failures. Nonetheless, also AI-based continuous monitoring introduces some challenges: AI works with data (tons of data in some cases) which should be somehow collected by means of (IoT) sensors. Here, there are two possible scenarios, either railway assets are already equipped by design with the adequate set of sensors allowing for real-time and/or massive data collection (e.g., those recently deployed), or they would required to be digitalised (e.g., those who have been operating for decades). In the second case, the costs to afford for the sensorisation may be a paramount challenge to be tackled. Furthermore, the installation of *intrusive* sensors, i.e., applied directly on or within the system to be monitored, could introduce additional complications in the case of safety-critical systems. These must be compliant with strict safety standards to be operational and modifications could induce costly and time-consuming re-approval processes. A possible solution, as evaluated in

this Thesis, would be to leverage *non-intrusive* and *cost-effective* sensors which could be installed externally to the assets without interfering with their operations.

1.1.4 Challenges for AI

As discussed, AI could open for different opportunities; nonetheless, it also raises several concerns, especially in terms of certifiability. Particularly, some AI techniques, including DL, do not yet seem to be mature enough to be directly welcomed by the strict safety standards regulating the railway sector, especially in the context of safety-critical applications [44]. There are some aspects related to the **trustworthiness** of AI systems, like *ethic*, *explainability*, and *human oversight* [38], that must be tackled before considering AI practically integrable in safety-critical (railway) systems. In addition to this, there are other, more technical and experimental, issues that are currently challenging the development of AI approaches, with **data availability and quality** being two of the most sensitive [138]. Data are nowadays considered valuable assets for (also railway [114]) companies, hence, it is understandable that their free disclosure would not be so strategic from the point of view of revenue; additionally, data sharing may be also held back by privacy and confidentiality concerns. Therefore, despite a few publicly available datasets exist, the majority of researchers have experimented AI approaches on data collected from scratch (and not published) or shared with them under disclosure agreement [138, 24].

1.2 Focus and Contribution

The discussion above sets the research context and the motivation for the work reported in this Thesis. *Attention is posed on the analysis of DL for railway safety and maintenance applications, specifically concentrating on solutions based on audio-video analysis; challenges and opportunities of DL are evaluated from both methodological and experimental perspectives with the aim of contributing to the future integration and adoption of DL in autonomous train driving systems and smart monitoring of safety-critical railway assets.*

The contribution of this thesis encompasses concepts systematisation,

methodological development, and experimental approaches as outlined herein:

- A novel *Reference Taxonomy for AI in Railways* (Chapter 2) is introduced with the principal objective of shedding light on the main concepts commonly attributed to AI (including DL) that would be of practical relevance for railway researchers and, above all, practitioners approaching the problem of integration of AI in railways.
- The concept of *Grade of Intelligence (GoI)* is introduced, which served as the basis for the *delineation of high-level guidelines for the step-wise integration of AI and DL in Autonomous Train driving* (Chapter 3). This vision also exploits the concept of *safety envelope*, introduced in Chapter 2, as a means to potentially mitigate and partially overcome current certifiability challenges.
- A *seven-step methodology systematising all the steps necessary for the development of DL applications* (Chapter 4) is presented. It has been outlined by leveraging well-established practices in the context of AI and DL, with the aim of designing specific guidelines for the railway community.
- Two case studies exemplifying the adoption of the seven-step methodology above are discussed. Specifically, *experiments were conducted and modular architectures proposed for vision-based Obstacle Detection (OD) on rail tracks* (Chapter 5) and *non-intrusive Level Crossings (LCs) monitoring through video/audio data* (Chapter 6).

These case studies were addressed with the aim of evaluating the opportunities introduced by DL in combination with: i) minimum on-board equipment for the detection of any possible obstacles on rail tracks; ii) non-intrusive and cost-effective sensors for the continuous monitoring of LCs.

Consequently, these case studies were taught as *benchmarks* for the investigation of the potentials and limits of DL in the context of some practical PoCs and under some pre-specified conditions. Therefore, while promising results were achieved, the implementation of advanced optimisation mechanisms towards the identification of the optimal solutions remains an objective for future developments.

- While addressing the case studies, one of the most prominent challenges was related to data availability. Then, with particular focus on the OD case study, it is discussed how synthetic data were generated through *the development of two virtual railway scenarios exploiting RoadRunner⁴ and Unreal Engine⁵*. To foster future research, the dataset generated through RoadRunner has been published on Zenodo⁶. Hopefully, this could serve as a benchmark for the development of future applications on this topic.

Worth highlighting, also some indications on how the dataset for warning bell classification (exploited in the context of the LC case study) was built have been provided on Zenodo⁷.

1.3 Structure

The rest of the Thesis is structured as follows:

- Chapter 2 provides the context for a better understanding of the Thesis, including the proposed taxonomy of AI and a review to place the research within the current scientific literature.
- Chapters 3 and Chapter 4 present the methodological framework in which the experimental contribution is placed. Chapter 3 introduces GoIs, the guidelines for the conceptual shift from *automatic* to *autonomous* trains, and an example architecture structuring the vision for full ATs in fully connected railway environments (GoI4). Chapter 4 contributes to systematising the steps generally necessary for the development of DL solutions into the seven-step methodology mentioned above.
- Chapters 5 and 6 are devoted to presenting the experimental contribution of the Thesis. They report on the approaches, DL models, and experiments conducted in the context of the two case studies.

⁴<https://it.mathworks.com/products/roadrunner.html>

⁵<https://www.unrealengine.com/en-US>

⁶<https://zenodo.org/records/7924875>

⁷<https://zenodo.org/records/7945412>

Chapter 5 addresses a specific OD problem, chosen for its crucial role in giving trains the ability to autonomously perceive threats in the environment. The case study investigates the detection of unknown obstacles on the rail tracks while leveraging data from a single RGB camera mounted on the head of the train. Chapter 6 addresses the health monitoring of LCs components for predictive maintenance purposes. This case study has been chosen given the relevance that LCs have in the railway domain, being the junction point between rails and roads, and also the stage of about one third of railway accidents. Hence, their operational integrity would be vital to ensure safety.

- Chapter 7 summarises the contribution of the Thesis and provides some conclusive remarks.
-

Chapter 2

Context and Background

This Thesis integrates two complex themes: *Deep Learning* (DL) and *railway systems*. The present chapter provides a high-level framework for both fields and contextualises the research described in the following chapters, introducing the necessary background for the topics that will be investigated.

To this aim, a taxonomy for AI in railway is presented, which is intended to disentangle and structure some of the aspects commonly associated with AI (those that were found to be relevant to the railway domain), and provide an overview of the DL-related aspects that have been of primary importance in the context of this Thesis.

Then, a high-level overview of current railway systems and practices is provided, focusing specifically on automatic train driving systems and maintenance activities emphasising, in the latter case, the role of non-intrusive approaches (based, e.g., on audio-video analytics).

Continuing, an analysis of the State-of-the-Art is presented. Given the focus on solutions based on audio-video analysis through DL (as anticipated in Chapter 1), such an analysis aimed at investigating how these solutions were adopted, and what opportunities they introduced, to innovate and/or support train driving systems and railway maintenance practices. Particularly, the topics of *vision-based obstacle detection on rail tracks* and *railway assets' monitoring through video and audio data* were considered.

Lastly, an overview of the main challenges to be considered when introducing DL in safety-related railway operations is provided.

2.1 A Taxonomy for AI in Railways

Artificial Intelligence encompasses a plethora of aspects, techniques and applications, making it essential to define a systematic framework that clarifies what AI represents, what its boundaries are, and what relationship exists between AI and other fields, before embarking on the development of AI solutions for a specific application domain. To this aim, this section introduces a taxonomy developed within the context of the research activities conducted in the railway domain and described in this Thesis. Worth emphasising, rather than being exhaustive, *the objective of the presented taxonomy is to shed a light on the main concepts that would have been of practical relevance for railway researchers and, above all, practitioners approaching for the first time to the problem of integration of AI in railways.*

As the first step, it is necessary to understand what an “intelligent” system is in the domain of interest and provide a pragmatic definition for AI that could be leveraged to distinguish between *intelligent* and *traditional* railway systems; this would also help to understand what benefits AI would bring in railway systems [8].

One of the most common definitions for *intelligent systems* is based on the *Turing test* [141], which elevates to *intelligent* those systems whose reasoning is indistinguishable from that of humans if evaluated by an impartial observer. Then, over the past years, different definitions of AI have been proposed (e.g., [1, 34, 109]), most of which share, to some extent, the following base concept: AI refers to a set of methodologies and techniques that provide systems with the capability of implementing behaviours commonly associated with intelligent human behaviours (e.g., learning from experience, or taking autonomous decisions to independently adapt to changes in the environment). Nonetheless, most of these definitions tend to broadly identify what AI represents and, thus, may result to be too abstract to be practically adopted in operational environments [8].

Therefore, without distorting the base concept of AI, in [8] it was provided a more pragmatic definition for AI that could be useful to railway practitioners:

“AI is the discipline gathering all the aspects that allow an entity to determine how to perform a task and/or take a decision based on the experience matured by observing samples and/or by interacting with an

environment, possibly competing against or cooperating with other entities.”

Where:

- *Aspects* include algorithms and techniques that would make systems capable of addressing tasks that would typically require intelligent human intervention.
- *Entity* refers to any possible kind of agent (e.g., a software or a train).
- *Experienced Maturity* includes both the concepts of *learning*, i.e., extracting knowledge from examples (or data), and *data-driven inference*, i.e., deducing consequences from priors.

This definition, evaluated within the rail sector, helps to discern next-generation intelligent railway systems from traditional ones implementing a programmed automatism [8]. An example is given by the current driverless trains implementing automatic train driving systems. These are capable of safely running unattended from one station to another leveraging a series of pre-defined rules and schedule [29], however, they are not capable of dynamically taking decisions to autonomously adapt to changes in the environment (e.g., react to obstacles on rail tracks) [8].

2.1.1 Reference Taxonomy

To further clarify what AI represents and how the various concepts attributed to it are related to each other, a new taxonomy for AI has been developed in the context of this Thesis work and proposed to the scientific and industrial community [8, 111]. It provides a formal systematisation of AI-related concepts based on UML class diagrams [75] (Figure 2.1).

As mentioned above, the taxonomy was not thought to be comprehensive, rather, it was conceived with the aim of clarifying those concepts that could immediately find practical feedback in the railway literature, e.g.: swarm intelligence algorithms in traffic planning and management applications [138], and computer vision and image processing, empowered by Machine Learning (ML) or Deep Learning, in assets defect detection [24] or perception systems [125]; ML and DL have actually been investigated in a plethora of railway applications [138].

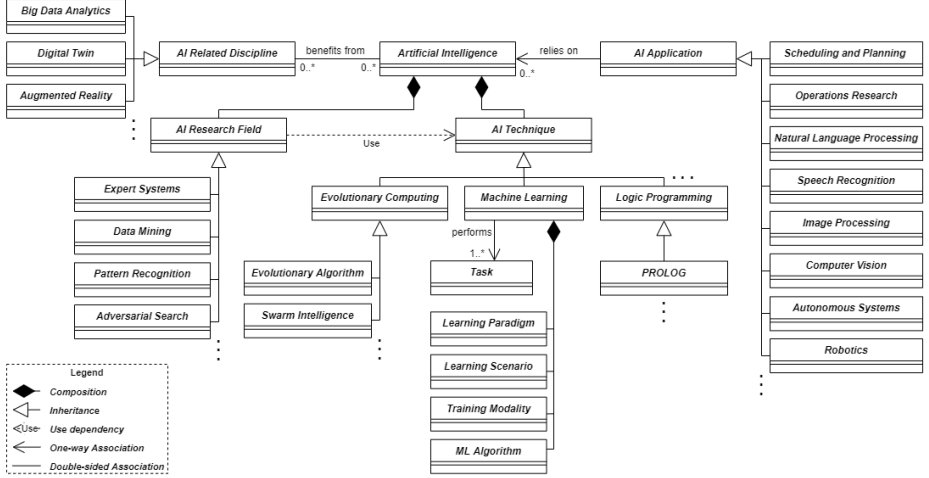


Figure 2.1. A Class Diagram for AI, excerpted from [111]

The main high-level concepts are represented by the *first level classes* and their relationships, i.e., AI Techniques, AI Applications, AI Research Fields, and AI Related Disciplines. Nonetheless, Figure 2.1 does not show the complete diagram: Machine Learning actually is a base class whose hierarchy is reported in Figure 2.2, while the three dots “...” indicate that the taxonomy can be further extended to encompass concepts and relationships not currently included. Also, worth emphasising is the absence of DL in Figure 2.1; this is mainly due to the fact that DL inherits from ML; a class diagram for ML is introduced below in this Thesis together with the contextualisation of DL.

Having stated that, an explanation for the class diagram in Figure 2.1 is given herein:

- The subclasses that inherit from **AI Technique** represent sets of algorithms and methods that are crucial for systems to be deemed “intelligent”. Indeed, such techniques provide systems with the capabilities of learning from experience, independently reasoning and selecting the actions to take, or implementing any other kind of behaviour typically associated with human/nature intelligence.

Machine Learning is one of the most known AI techniques. Particu-

larly, by recalling the definition that Tom Mitchell associated with it, “*Machine Learning is the study of computer algorithms that improve automatically through experience*” [96], from a high-level perspective, ML techniques can be considered as those that give systems the ability of autonomously learning how to perform a specific task without being specifically programmed to address it.

- **AI Application** identifies research areas, domains, topics, and so on, that are independent from AI but whose performances could be (largely) improved if AI techniques are involved.

Some of these AI applications are relying so much on AI that, in some cases, are considered to be feasible only with AI techniques; which is not properly correct. A striking example is given by *Computer Vision*, which, by performing trivial research on the internet, is classified, in some cases, as an “AI discipline”. However, there is a plethora of Computer Vision (CV) approaches that do not rely on any AI technique; AI, especially in the form of DL, is serving as a means to (largely) improve some of traditional CV practices [101].

- Differently from subclasses that inherit from AI Application, those specialising **AI Research Field** represent disciplines, domains, and research areas that were born with AI and directly depend on AI techniques; hence, the adoption of AI is not optional or a way to improve performances, instead, AI is the core of the research field itself.

Examples of AI research fields, among others, are *Expert Systems*, which can be seen as the branch of AI focusing on the realisation of systems capable of emulating the decision processes made by some experts in a specific application area [15], and *Data Mining*, which includes procedures oriented at extracting useful information from raw data [152].

- Lastly, **AI Related Discipline** specialises into particular domains which could support AI and, at the same time, benefit from it.

A striking example is given by Digital Twins (DTs), which can be defined as digital representations of physical assets which are connected to them by means of (IoT) sensors and evolve with them over time

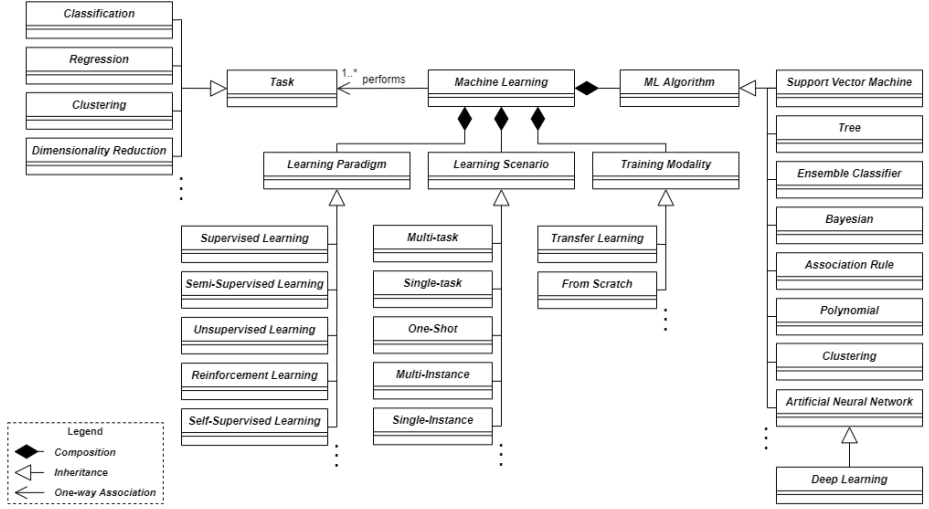


Figure 2.2. A Class Diagram for ML, excerpted from [111]

[23]. DTs can support AI in *data generation*: for example, injection approaches [103] can be adopted to generate synthetic data with which train and test AI models. On the other hand, AI could *extend DT functionalities* from data pre-processing to the deployment of intelligent services (e.g., predictive maintenance) [23].

2.1.2 From Machine Learning to Deep Learning

The ML hierarchy is depicted in Figure 2.2. This was realised by considering the following rationale [111]: *Machine Learning accomplishes a given Task through the adoption of a particular (ML) Algorithm that is trained using a specific Learning Paradigm within a defined Learning Scenario and taking into account a predetermined Training Modality.* Where:

- **Task** represents the goal that is intended to be pursued, e.g., *Classification*, i.e., the problem of associating a specific instance with the right class.
- **ML Algorithm** represents a sequence of operations used to train a specific model. Among the others [152, 9], Artificial Neural Networks

are of particular interest for this Thesis since [DL](#) (introduced below) directly inherits from them.

- **Learning Paradigm** refers to the strategy that guides the algorithm during the training process.

Among others, *Supervised*, *Unsupervised*, and *Semi-Supervised* Learning [152] are of primary importance in the context of this Thesis. The first one expects to accomplish a specific task while leveraging data which have been previously labelled; conversely, the second one, aims at addressing a task while extracting information from data which have not been labelled. Lastly, Semi-Supervised Learning conceptually combines the two above: a task is tackled by exploiting data of which only a subset is labelled.

- **Learning Scenario** describes the characteristics of the *Task* that is intended to be addressed.

Examples are: *Single-Task* versus *Multi-Task* [14], where, conceptually, the latter, in contrast with the former, expects to solve multiple tasks at the same time through the application of a single *ML Algorithm*; and *Single-Instance* versus *Multi-Instance* [152], where, conceptually, in the first case, each instance represents a single event, while, in the second case, instances are grouped in “bags” where each bag represents a specific event and each instance of the bag represents the same event but from a different perspective.

- **Training Modality** indicates how the training is implemented.

From a high-level perspective, among others, two main *modalities* exist: *From Scratch* and *Transfer Learning (TL)* [104], both of them play a key role in the context of this Thesis. *TL* expects to re-use the knowledge acquired to solve a specific task (source task) in a specific domain (source domain) to address a similar or different problem (target task) in the same or another application field (target domain). Differently, in training *From Scratch*, the *ML Algorithm* is directly trained on the collected data without leveraging any pre-acquired knowledge. The main advantage in applying *TL* lies in the fact that it would be possible to obtain competitive results even with a limited amount of data [104]. Additional details are given below.

In this vast context, as partially mentioned above, Deep Neural Networks (DNNs) can be seen as a particular subset of Artificial Neural Networks with a deeper structure, which translates into more layers and parameters. In addition to that, the peculiarity of DL compared to “traditional” ML lies in the automatic feature extraction [76]. Basically, while in ML the user has to perform some *feature engineering* selecting those that could potentially be the most representative of the problem, in DL, relevant features are automatically extracted (actually, learned) by the DNN itself. This has brought many advantages in the analysis of raw, high-dimensional data including images, text, and audio [76].

A class diagram similar to the one shown in Figure 2.2 could be generated for DL, however, most of the concepts would remain the same. It is worth understanding, instead, how TL can be applied in the specific context of DL, also known as Deep Transfer Learning (Deep TL) [66, 136], which has been of particular interest for the research discussed in the following chapters.

Each DNN has a series of internal parameters, named weights, which update during the training phase. Typically, in the case of *training from scratch*, these weights are initialised with some random values [152]. Alternatively, these weights could be initialised with values obtained while learning how to solve a previous (source) task; this conceptually means transferring the knowledge from the source task to the target task. The weights that have been obtained by training a DNN on a specific set of data (and that are intended to be used as transferable knowledge) are referred to as “pre-trained weights”; the DNN, similarly, takes the name of “pre-trained model (or network)” [66]. Among the various approaches [66, 136], in this Thesis, the focus has been mainly put on *Network-based Deep TL*, which expects of re-using part or the totality of a pre-trained network, including its structure [136], to solve a specific task.

Figure 2.3 graphically shows how Deep TL can be applied in the case of a Convolutional Neural Network (CNN) (which is a particular class of DNN [152]) used to perform a classification task. Basically, the network can be seen as composed of two parts, some convolutional layers, which perform the *feature extraction*, and some (typically) fully-connected layers, which perform the actual classification [66]. In this context, there could be two main scenarios, both of which expect to re-use at least the pre-trained

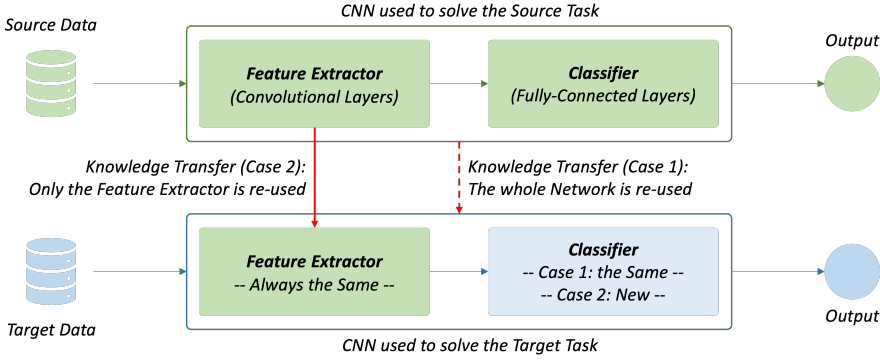


Figure 2.3. Examples of Transfer Learning

weights of the feature extractor [66, 136]:

- **Case 1: The whole network is re-used.** All the pre-trained weights of the whole pre-trained network are exploited. Then, it is possible to proceed in two main ways:
 - *Freeze the pre-trained weights of the feature extractor and Fine-tune those of the classifier.* Where, *freeze* means that, during the new training phase on target data, the weights are not allowed to update; conversely, *fine-tuning* means that the weights are allowed to evolve during the new training.
 - *Fine-tune all the pre-trained weights.*

For all those who were wondering, the third “logical” option (i.e., freeze all pre-trained weights) is not exactly **TL**. Target data are not adopted to “update” the network’s knowledge and it would simply mean to re-use a pre-trained network to try to infer something related to the target task without modifying it in any manner. This practice is mainly adopted when building a **DNN**, i.e.: it is trained on some data leaving the weights (or part of them) free to update, then, once the training is finished, all weights are frozen and the network is used in “inference mode”. A very trivial example is the following: if a network has been trained to recognise cars, it would most likely fail

to recognise motorcycles if not re-trained (whether adopting TL or not) on motorcycle data.

- **Case 2: Only the feature extractor is re-used.** This basically means dropping all the fully-connected layers of the pre-trained network and substituting them with some other layers that would be more appropriate for the target task. Then, similarly to the previous case, the pre-trained weights of the feature extractor could be both *frozen* or *fine-tuned*.

The concepts above can be generalised also for other types of DNN, like Autoencoders [4] or more complex architectures like those specifically designed for object detection (e.g., YOLOv3 [120] or Faster R-CNN [122]); for example, in these cases, all pre-trained weights can be loaded and fine-tuned. Deep TL largely contributes to reducing the training time (on the target task) and obtaining competitive performance even if only limited target data are available [66], as also demonstrated in the following chapters.

To conclude, it is necessary to underline that, in this Thesis, Deep TL was adopted for scientific purposes to understand its potential in the context of the two analysed case studies. However, when developing DL systems to be used in operational railway environments, there are some important aspects that would be essential to consider including, among others, *patent rights*, that should be evaluated for the commercial adoption of pre-trained models, and the *validation and verification* of pre-trained knowledge and/or source data, to ensure that they are reliable, unbiased, and, in general terms, trustworthy (especially in the case they would be adopted for safety-critical applications).

2.1.3 Deep Learning for Computer Vision and Audio Processing

To conclude with the examination of AI, ML, and DL concepts, it is worth underlining the relationship that, according to what has been discussed so far, holds between DL and AI Applications, especially Computer Vision and Audio Processing which represent the main focuses of the experimental part of this Thesis (Chapters 5 and 6).

As for Computer Vision, a huge number of DNNs, especially CNNs, have found application in several contexts, laying the ground for the technological advancement in different safety-related domains, including health care [82], automotive [106], and railways [125, 24].

Among the plethora of DNNs proposed for video and image analysis, there are a few specific DL architectures that have introduced such innovation in the field and have proven to be so effective that they have become a sort of standard or an extremely efficient starting point for solving specific tasks. A few examples are: VGG [131], a CNN that has found application especially in image classification, but has also inspired the realisation of or has been re-used in more complex DL architectures (e.g., [122]); YOLO (e.g., [119, 120]) and Region-based CNN (e.g., [122]) families, which have demonstrated to be effective for object detection tasks; and SegNet [3] and U-Net [127], which have been largely adopted for Semantic Segmentation (SS).

The same holds for Audio Signal processing, where DNNs have been widely investigated to deal with, among others, speech, music, and environmental sound analysis addressing tasks including automatic speech recognition, and audio classification, detection, and generation [110].

The experimental research activities reported in the following leverage Semantic Segmentation, Anomaly Detection, (implemented through Image Reconstruction), Audio Classification, and Object Detection; therefore, some background information is provided to facilitate the reading of the case studies:

- Semantic Segmentation aims at associating each pixel within an image with a semantic label. The application of DL to this task typically expects to train a DNN, e.g., U-Net, to predict a “mask” for the input image highlighting the object that is intended to be segmented; from a high-level perspective, the mask is nothing more than another image whose pixels’ values represent the classification for the corresponding pixel in the input image. The usage of SS and U-Net is reported in Chapter 5 to segment rail tracks.
- *Anomaly Detection through Image Reconstruction* can be defined as an approach to detect anomalies within an image by leveraging DNNs, such as Deep Autoencoders or their variants (e.g., Vector Quantised-

Variational Autoencoder (VQ-VAE) [144]), trained in unsupervised mode and oriented at reconstructing the image in input. A possible reasoning would be: if the DNN is trained on non-anomalous data only, at run-time, it would not be capable of properly reconstructing possible anomalies. An anomaly map is then computed by comparing the image in input with its reconstruction. The usage of this approach is reported in Chapter 5 for the detection of unknown objects on rail tracks.

- Audio Classification (intended as acoustic scene classification [110]) refers to the problem of associating an audio recording with a specific label. Such a task can be addressed by means of CNNs oriented at analysing visual representations of the audios like spectrograms (e.g., [129]). Audio Classification was taken into account to perform warning bells detection (through classification) in the context of level crossings, as explained in Chapter 6. A particular VGG-like architecture, named VGGish [64], was leveraged, which was originally conceived to extract features from AudioSet [64] data.
- *Object Detection* refers to the recognition and localisation of objects in images. Among the available DL architecture designed for this task, YOLO and Region-based CNNs are among the most exploited. The former are usually faster, while the latter typically achieves higher detection accuracy [83]. These DNNs, as the others of this kind, are named Object Detectors and are oriented at predicting “bounding boxes” (i.e., rectangles) each of which contain one of the object that has to be detected in the image; then, also the class for each object is predicted. The usage of an Object Detectors, and specifically YOLOv5¹, is reported in Chapter 6 where it has been exploited to predict the bounding box containing the level crossing protective barrier in adjacent video frames. The heights of the bounding boxes detected over time have then been plotted and used to characterise the movement of the barrier.

¹<https://github.com/ultralytics/yolov5>

2.2 Overview of Relevant Railway Concepts and Systems

This section provides a high-level overview of the railway concepts and systems which are relevant to the work carried out in this Thesis. Hence, it introduces the systems and practices currently adopted in railways in the context of automatic train driving and some notions that would be essential to understand the shift towards Autonomous Train (AT) and smarter (and non-intrusive) monitoring of railway assets.

2.2.1 Automatic Train Driving Systems

Over the past few years, there has been an increasing interest in automating railway lines in order to optimise their capacity and efficiency. At the current stage, railway lines are classified according to five Grades of Automation (GoAs) [67] reflecting the presence or absence of the *automatic train systems* summarised in Table 2.1 and the need for human operators (i.e., driver and crew members) ensuring safe operability [44]. The five GoA levels are summarised herein:

- **GoA0:** *On-sight train operation.* No automation is present and train operations are supervised by the driver.
- **GoA1:** *Non-automated train operation.* Train operations are manually managed by the driver supported by the ATP.

Table 2.1. Automatic Train Systems, excerpted from [44]

System	Description
Automatic Train Operation (ATO)	Automatically drive the train and stop at stations when needed.
Automatic Train Protection (ATP)	Automatically applies protection mechanisms (e.g., emergency brakes) if required.
Automatic Train Control (ATC)	Both ATP and ATO are in place to automatise train driving and ensure safety.
Automatic Train Supervision (ATS)	Automatically manages train schedule and coordinate routes along whole tracks.

- **GoA2:** *Semi-automatic train operation.* ATO and ATP automatically manage trains operations while supervised by the driver.
- **GoA3:** *Driverless train operation.* Train operations are completely automatic and driverless, while crew members handle possible emergencies.
- **GoA4:** *Unattended train operation.* Train operations are completely automatised, no human operator is required.

Other important concepts that are worth highlighting in this context, as anticipated in Chapter 1, are those of:

- **Rule-based (or Regulated) Safety and Managed Safety** [124]. The former aims at defining proper countermeasures to known and predictable hazardous events by means of, indeed, rules, formalisms, and protection mechanisms; the latter aims at mitigating and possibly avoiding non-predictable hazard events.

According to these interpretations, *Rule-based Safety* can be ensured by means of systems like the ATP, while *Managed Safety* is endorsed by human operators who react, at run-time, to situations and events that cannot be handled by means of pre-specified rules.

- **Segregated and Open Railway Lines** [95]. The former category encompasses lines that are completely isolated from external threats (e.g., metro lines with platform screen doors), while the latter includes lines that present points, e.g., level crossings or stations, from where external agents could (easily) access rail tracks.

Up to date, GoA4 has been achieved in some *segregated* lines, where trains operate completely unattended with high efficiency². On the other hand, in *open* lines, full automation (i.e., GoA4) seems yet to be achieved due to the high interoperability requirements posed by such large, complex interconnected railway networks [44]. Different steps have been taken over the past few years towards the introduction of ATO in *open* lines. The so-called ATO over ETCS (AoE) – i.e., ATO supervised by the European

²<https://www.uitp.org/topics/automated-metros/>

Train Control System (ETCS) which manages all safety aspects – has been recently tested and prototyped in Europe; interestingly, in addition to that, there have been carried out tests evidencing that ATO functionalities may also be introduced in non-ETCS railway lines [130]. It seems that these functionalities would allow for upgrading *open* lines up to GoA2 [130] where, in the case of AoE, the *rule-based safety* would be ensured by the ETCS, while the driver will supervise the functioning on the automatic systems ensuring what was defined above as *managed safety*. Worth underlining, specifications for AoE up to GoA3/4 have already been drafted in the context of the Shift2Rail project X2RAIL-4 [153], and further investigations are being carried out by other European projects like Europe’s Rail FP2-R2DATO³.

Among the subsystems required to perform the paradigm shift from GoA2 to GoA3/4 in *open* lines according to [153], this Thesis focuses on the **perception** subsystem, i.e., the one that is oriented at analysing the environment to understand whether the train’s run could proceed safely or actions would be required to mitigate or avoid possible threats. To be specific, one of the focuses of this Thesis, as introduced in Chapter 1, is to evaluate the potential of DL-based CV approaches when it comes to detecting obstacles on rail tracks. To that aim, in Section 2.3, an overview of the latest scientific developments in this direction is provided; then, Chapter 3 discusses additional concepts and methodologies related to the introduction of AI and DL in train driving systems, envisioning a possible conceptual shift from automatic to autonomous trains.

2.2.2 Maintenance and Audio-Video Analytics

Maintenance and system safety are related concepts. Railways encompass a number of safety-critical assets whose correct functioning is crucial for safe operability. From a conceptual perspective, maintenance refers to all activities intended to keep railway assets and systems in adequate operating condition with the aim of, among others, avoiding possible consequences on safety due to improper functioning [112].

As anticipated in Chapter 1, maintenance and inspection activities are traditionally conducted on the field by human operators, following

³<https://projects.rail-research.europa.eu/eurail-fp2>

specific procedures and, in some specific cases, with the support of special instruments and/or vehicles [24]. Additionally, inspections are usually conducted according to a fixed schedule and when the traffic is suspended (e.g., during the night). These activities, despite having been standards for decades, might result to be extremely expensive, in terms of both sensor costs and required time, and could not allow for the timely detection of anomalies: malfunctions occurring between adjacent inspections might not be detected in time. These are among the main motivations that pushed researchers and practitioners worldwide to investigate approaches that could allow the paradigm shift from *scheduled and corrective* maintenance to *continuous and predictive* maintenance. The incremental adoption of technologies allowing for real-time and massive data collection, e.g., **IoT**, and the spread of advanced data processing approaches, like **ML** and **DL**, are largely supporting this shift [22].

In this context, one of the main challenges regards the sensorization and digitalisation of railway assets. The newest assets may be already equipped with the latest sensors and technologies allowing for real-time data collection and continuous monitoring, nonetheless, those that have been operational for decades might not. In the latter case, the sensorisation and digitalisation of the assets would require massive costs and time. Additionally, in case of safety-critical assets, they must be compliant with strict safety standards in order to be deemed operational (e.g., CEN-CENELEC standards for the assessment of the Safety Integrity Level (**SIL**) [128]); physically installing sensors on or within these assets might invalidate their compliance and lead to expensive and time-consuming re-approval processes [25].

In such a context, audio and video sources could introduce several advantages including:

- *Non-intrusiveness.* They can be installed externally to the asset to be monitored, without interfering with either their compliance with safety standards or their operation.
- *Technology re-usability and cost-effectiveness.* In some specific cases, e.g., at level crossings, sensors like cameras and microphones (often part of smart cameras) might be already installed for other reasons (e.g., safety/security surveillance); These sensors could be used also for maintenance purposes saving additional installation costs.

On these bases, the research activities conducted within this Thesis work addressed DL applications for railway maintenance based on audio and video data and included a review [24] whose findings are reported in Section 2.3.2.

2.3 A Review of DL Solutions Enhancing Railway Safety and Maintenance Applications

Deep Learning has been widely investigated in recent years to cope with, among others, railway automation, maintenance, safety, and traffic planning and management applications [138]. Different solutions have been proposed, covering applications like defect detection [24], failure prediction, energy consumption optimisation, and trains delay prediction [138].

In this vast scenario, as underlined in Chapter 1, this Thesis mainly focuses on how DL can be applied to improve safety in ATs and support maintenance/inspection activities if only non-intrusive sensors (cameras and microphones) are involved. Specifically, in the context of ATs, attention is posed on vision-based systems that could support the technological advancement towards *driverless* trains in *open* environments by detecting obstacles on rail tracks. Hence, this section provides an overview of the State-of-the-Art in the following directions: “Vision-based Obstacle Detection (OD) on Rail Tracks” and “Railway Assets Maintenance through Video/Audio Data”.

2.3.1 Obstacle Detection on Rail Tracks

Among the subsystems required to perform the paradigm shift from GoA2 to GoA3/4 in *open* lines according to [153], the **perception** system would be in charge of analysing the environment (*“in place of the driver”* [153]) to, among other functionalities, detect obstacles that could affect train’s safety. Therefore, OD would be an extremely relevant functionality for the implementation of driverless trains in *open* environments.

Obstacle Detection systems typically involve the utilisation of multiple sensors like RGB and thermal cameras, radars, and LiDARs [125, 135]. The need for this multitude of sensors principally comes from the fact that each of them could help to overcome the shortcomings of the others. For

example, cameras may suffer from particular weather and illumination conditions, while LiDARs allow for robust distance estimation also during nighttime [135]; at the same time, *cameras are the only sensors that could provide high-resolution visual representation of the surrounding environment* [125], *allowing for objects' recognition* [125, 135], i.e., understanding the type/class of the obstacles.

Important to mention, in the context of this Thesis, the concept of Obstacle Detection combines two factors: *obstacle recognition*, introduced above, and *obstacle localisation*, i.e., understanding where the obstacle is located within the environment. With the advent of DL and the plethora of CV approaches (whether based on AI/DL or not), cameras, although not free from shortcomings, have become essential in multi-sensor OD systems [125] as through simple images it would be possible to both *recognise* and *locate* obstacles in railway environment. In addition to that, cameras are cheaper than other sensors and can also be used for multiple purposes, e.g., monitoring rail tracks or catenary defects (as better discussed in Section 2.3.2).

Inspired by what the authors in [125] wrote about the types of OD systems, and taking into account the various solutions proposed in the literature for *vision-based OD* on rail tracks (discussed below), OD systems can be classified into two main categories:

- *Fixed*: OD systems installed as trackside equipment which are, indeed, not movable and are in charge of monitoring pre-specified railway areas.
- *Dynamic*: OD systems installed on-board running vehicles which, taking into account the various approaches proposed in the literature, could be further subdivided into:
 - *On-train*: OD systems installed on trains which monitor rail tracks ahead the trains and the surrounding environment.
 - *Air-born*: OD systems installed on drones monitoring the rail tracks and the environment from a different perspective.

Figure 2.4 graphically structures this classification. These OD systems are not mutually exclusive, rather, they would be complementary [125]:

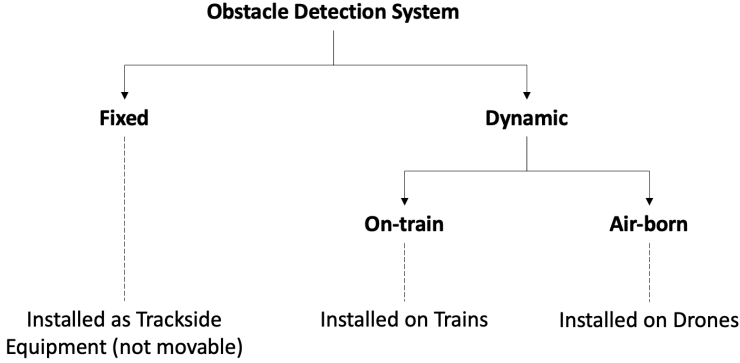


Figure 2.4. A High-Level Classification of Obstacle Detection Systems

the *On-train* would be the primary OD system which dynamically and constantly monitors tracks ahead ATs, while *Air-born* and *Fixed* OD systems would monitor areas that could not be properly monitored through the *On-train* OD system only (e.g., in curves, where trains' visibility is limited) or that are particularly critic (like level crossings) to achieve more efficient and long-range obstacle detection [46].

Premised that *sensor fusion* and the integration of *Fixed*, *On-train*, and *Air-born* OD systems would be essential for robust OD, in this Thesis, attention is specifically posed on **vision-based On-train OD systems supported by DL**. This focus is also motivated by the central role that cameras have in object recognition as discussed above and by the cost-effectiveness of such sensors. Indeed, there is a plethora of railway lines which are not automated or which are not assisted by any protection mechanisms where vision-based OD systems could play a key supportive role to affordably improve safety [44].

Vision-based Deep Learning Applications for On-train Obstacle Detection

In early 2021, an in-depth review of *vision-based DL applications for On-train OD* was proposed in [125]. The authors analysed several studies adopting *traditional CV techniques* (like edge/corner detection and threshold segmentation techniques not leveraging AI/DL), *DL-aided CV*

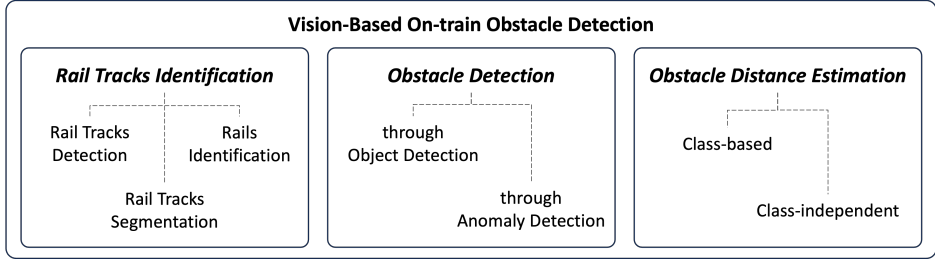


Figure 2.5. A Classification of Approaches for Vision-based On-train OD

techniques, or their combinations. Among the various interesting aspects discussed in that review, the most relevant for this Thesis work are summarised herein:

1. Traditional CV approaches are more prone to be affected by changing environmental conditions (e.g., lights, weather, and backgrounds) than DL approaches. The latter, differently from the former, could autonomously learn how to cope with these changes.
2. On-train OD systems typically involve three main modules namely, *Rail Tracks Identification*, *Obstacle Detection*, and *Obstacle Distance Estimation*. The first two were largely investigated, while only limited work was found for Obstacle Distance Estimation.
3. Most of the reviewed studies dealing with Obstacle Detection, investigated supervised DL approaches.

This picture does not seem to have changed significantly in the last few years. By reviewing the recent literature, it was found that scientific studies have mainly focused on the same three modules indicated above. Then, they tried to implement those by adopting different approaches that can be clustered as depicted in Figure 2.5.

Approaches for Rail Tracks Identification. These approaches are oriented at identifying rail tracks (or only rails) within images. They can be subdivided into: *Rail Tracks Detection*, *Rail Tracks Segmentation*, and *Rails Identification* approaches. Figure 2.6 graphically shows the difference between them:

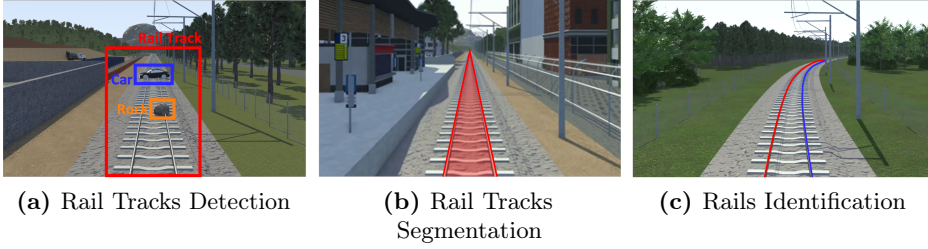


Figure 2.6. Examples of Rail Tracks Identification Approaches
(Images from RoadRunner)

Rail Tracks Detection approaches consider rail tracks as objects to be identified within the images, simultaneously with objects (potential obstacles) standing on/near rail tracks. Hence, studies proposing these kinds of approaches mainly focused on the development of DNNs capable of detecting objects at different scales (large, but particularly, small objects) [159, 160, 161, 162]. Some of these were inspired by or adapted well-established DNNs like YOLOv3, ResNet [63], and Single-Shot Detector [84].

Rail Tracks Segmentation approaches aim at segmenting the area occupied by rail tracks (including, part of the ballast, sleepers, and rails) within the images. Examples are given in [19, 147, 149], where the authors implemented DNNs for SS by re-using or taking inspiration from BiSeNet [163], ResNet50 [63], and SegNet [3] respectively.

Rails Identification approaches focus on separately detecting/segmenting rails (instead of detecting/segmenting rail tracks as a whole). Two main kinds of approaches were investigated: key-points detection [150, 168] and Semantic Segmentation [68, 87]. Worth mentioning, the authors in [168] started from the assumption that detecting rails is quite similar to detecting road lane markings; then, they adapted the PINet architecture proposed in [73] for lane marking detection. Similarly, reference [87] presents a DNN based on LaneNet [100], originally conceived to detect road lanes.

For the sake of completeness, it is also worth mentioning the work

performed in [60], which is halfway between *Rail Tracks Detection* and *Rail Tracks Segmentation*. Indeed, the authors leveraged Mask R-CNN [62] to simultaneously perform both object detection and semantic segmentation. In this particular case, “object” refers to both obstacles and rail tracks.

Approaches for Obstacle Detection. Two main strategies were identified in this context. To properly understand them, it is necessary to clarify the following aspect. Throughout this Thesis, when discussing aspects related to Obstacle Detection, the term **object** refers to obstacles that are known and are pre-specifiable, while the term **anomaly** refers to the obstacles that have not been or could not be pre-specified. Then, the term **obstacle** refers to both objects and anomalies indistinctly. Having highlighted that, the two main strategies that were found are:

Obstacle Detection through Object Detection. This strategy expects to leverage Object Detectors trained in *supervised mode*. It allows for both *obstacle recognition* and *localisation*, but its applicability is limited to pre-specifiable and limited objects only. Approaches of this kind have been largely proposed in the literature.

First, it is worth underlining that the studies mentioned under *Rail Tracks Detection* also perform object (intended as known obstacles) detection. Then, the plethora of studies adopting such a strategy can be clustered according to the State-of-the-Art DNN they took inspiration from.

A number of works leveraged YOLOv3 [126, 78, 43, 21], YOLOv4 [71, 61, 55], and YOLOv5⁴ [13] by re-using, adapting, or improving them. Particularly, reference [126] discusses some of the results achieved within the Shift2Rail project SMART⁵ where researchers experimented a complex multi-sensor system (including RGB and thermal cameras) for OD on rail tracks, proposing also approaches for vision-based *Obstacle Distance Estimation* (discussed below).

Besides YOLO networks, other State-of-the-Art DNN were leveraged. These include Faster R-CNN in [51, 164], Inception-v3 [133] in [20],

⁴<https://github.com/ultralytics/yolov5>

⁵https://projects.shift2rail.org/s2r_ip5_n.aspx?p=SMART

CenterNet [166] in [46], and Single-Shot Detector and ResNet in [155]. Notably, reference [46] presents some results achieved within the Shift2Rail project SMART2 (successor of SMART) and the same DNN is used to analyse images from both the on-board cameras and cameras mounted on drones, which are used to support the on-train OD system.

To conclude, reference [70] proposes a comparative study evaluating the performance of different object detectors (YOLOv2 [119], YOLOv5, Faster R-CNN, and Single-Shot Detector). According to their tests, Faster R-CNN achieved the highest accuracy but was also the slowest, while YOLOv5 had the best trade-off between accuracy (slightly worse than Faster R-CNN) and computational time (much lower than Faster R-CNN).

Obstacle Detection through Anomaly Detection. Unlike the above strategy, this expects to train DNNs in *unsupervised mode* and the aim is to identify any kind of anomalies. Despite the number of detectable obstacles is potentially larger, this strategy only allows for *obstacle localisation* as no class is associated with the detected anomaly.

Such a strategy has received only limited attention so far, with solutions being published especially in recent years. References [68] and [11] proposed approaches based on Deep Autoencoders. In both cases, the networks were trained to reconstruct the image in input and only non-anomalous images (i.e., without obstacles) were adopted. Then, at run-time, the (potential) anomaly would be poorly reconstructed and, thus, identified. In [68], the focus has been posed on detecting anomalies laying exactly on rails; while in [11] the anomaly is identified by evaluating a “gap-score” between the reconstruction error when the anomaly is present with the reconstruction error computed on “reference” images depicting the same scene but without anomalies.

The authors in [12] proposed two approaches. The first leverages a *Rail Tracks Segmentation* approach according to the following rationale: if the SS network is trained with non-anomalous images only to detect rail tracks, a possible anomaly would be classified as the background. Then, similarly to [11], the segmentation mask can be compared with that obtained from a “reference” image depicting the same

scene but without anomalies. The second approach, instead, adopts an image reconstruction method based on Generative Adversarial Networks (GANs) trained on non-anomalous images. In case an anomaly is present on the rail tracks, it would be poorly reconstructed, therefore, the anomaly can be highlighted by comparing the image in input with its reconstruction by means of another DNN (for which, not enough details seem to be given).

Lastly, in reference [86], the authors proposed an approach combining a DNN to extract features and an NFlow-based method to localise anomalies; basically, they first estimated the density distribution of pixel-level features extracted from non-anomalous images, then, applied the normalising flow approach to localise anomalies based on the density distribution.

To conclude with OD, reference [48] seems to be the only one combining (to some extent) both the strategies. A Deep Autoencoder (trained with non-anomalous images) is used to reconstruct the image in input; then, some “difference images” are obtained by subtracting the original image and the reconstructed one in two different ways; these difference images are then stacked and put in input to a CNN that classifies them in non-anomalous or anomalous; in case of anomaly, another Deep Autoencoder is triggered, which localises the anomaly within the image; lastly, the anomalous area is cropped out and processed by another CNN that predicts the class of the object.

Approaches for Obstacle Distance Estimation. Also in this case, the approaches can be subdivided into the two categories discussed herein:

Class-based approaches require to know the class of the object for which it is intended to estimate the distance. Consequently, these can only be used in combination with *Obstacle Detection through Object Detection* approaches.

A total of three approaches for *Obstacle Distance Estimation* have been found, two of which can be catalogued as *class-based*. Both of them have been developed in the context of the SMART project, where two networks, named DisNet [58] and its improved version

Multi-DisNet [59], have been implemented for this purpose. The main differences between the two are: DisNet analyses one image at a time, but it is possible to estimate the distance of different types of objects within the image; Multi-DisNet, instead, simultaneously processes two images (one from a RGB camera and the other from a thermal camera) depicting the same rail scene, but it seems to be capable of focusing on a single object class “at the time”. Both the networks were trained in *supervised* mode: in the case of DisNet, samples were labelled with the distance from the train measured through a laser scanner; in the case of Multi-DisNet, the dataset has been generated by acquiring images depicting obstacles at X meters from the train. The functioning is similar: an image is processed by an Object Detector that predicts the bounding boxes for the objects contained in the image; those bounding boxes are then presented in input (together with the class in the case of DisNet) to the distance estimation network that, indeed, estimates the distance.

Class-independent Approaches do not require to know the class of the obstacle to estimate the distance. These can then be used in combination with both OD strategies.

Only a single study [107] was found for this category and, actually, it proposes an approach that, depending on the specific configuration, can be classified either as *class-based* or as *class-independent*. It combines a homography-based distance estimation approach with a Multilayer Perceptron⁶. Independently from the specific configuration, the approach takes in input information related to the bounding box surrounding the object (which can be drawn also for anomalies by applying a trivial post-processing algorithm). As highlighted by the authors, one of the main challenges concerns the relation between the bounding box and the estimated distance; to small variation in the prediction of the bounding box may correspond huge variation in the estimation of the distance. Also, if the lower edge of the bounding box does not intersect with rail tracks, the distance estimation may be

⁶Technically, the Multilayer Perceptron does not fall under DL; however, the proposed approach is worth mentioning as it seems to be the only of its kind (i.e., possibly *class-independent*)

highly inaccurate. Only a specific configuration (named MLP3, the *class-independent* one) seems mitigating such a problem, producing accurate approximations according to the performed tests.

Review Findings and Discussion. A wide number of studies focused on *Rail Tracks Identification*, *Obstacle Detection through Object Detection*, or a combination of both. Then, a very few studies (3) added a *Class-based Obstacle Distance Estimation* approach to this pipeline. Similarly, also *Obstacle Detection through Anomaly Detection* seems to have been poorly investigated.

Most of *Rail Tracks Identification* and *Obstacle Detection through Object Detection* approaches work with high performances within the conditions set (including the datasets) by the authors. However, despite some studies performing comparative analyses to evaluate their solution against other (State-of-the-Art) DL models, specific evaluation methodologies would be required (e.g., by establishing the same evaluation criteria and considering the same dataset) to comprehensively evaluate the effectiveness of all the proposed applications. Hence, it is not straightforward to indicate which specific DL model would perform the best under all possible conditions, instead, some assumption can be made on the type of approach that should be adopted.

Particularly, to build a “stand-alone” vision-based OD system, all the modules above would be required. Also, the *Obstacle Detection* and the *Obstacle Distance Estimation* modules should be implemented by adopting both corresponding approaches. Basically, by combining the potentials of all of them, it would be possible to define a vision-based OD system that (visionary) could be capable of *locating* all possible obstacles while keeping the main advantage that cameras introduce (i.e., visual object *recognition*) for a specific set of sensitive and most recurrent potential obstacles (like humans or vehicles). A modular approach of this kind will be discussed in Chapter 5, where the attention will be posed on the implementation of a *Rail Tracks Segmentation* module and an *Obstacle Detection through Anomaly Detection* module, given: i) the limited attention that the latter has received in the literature so far; and ii) the fact that, according to the considered strategy, the former is propaedeutic to the latter.

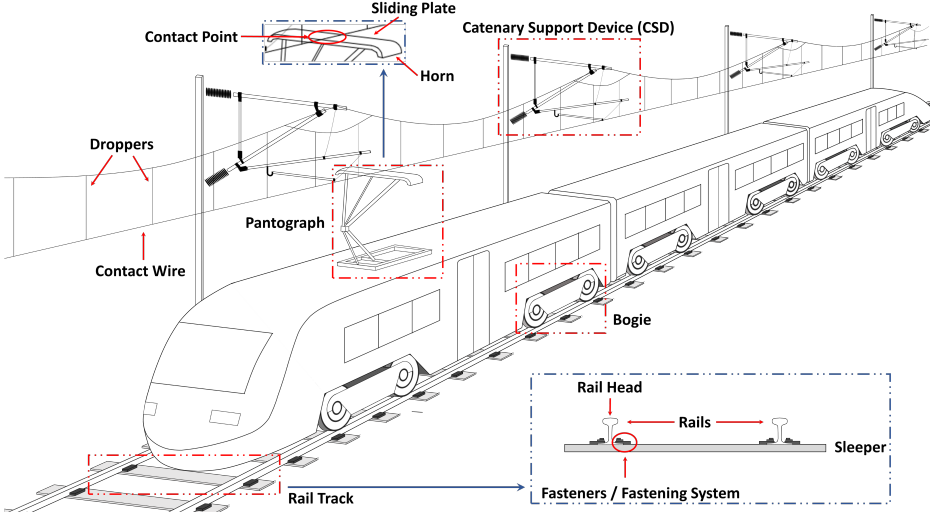


Figure 2.7. Overview of Some Railway Relevant Assets and Related Components [24]

2.3.2 Assets Monitoring through Video/Audio Data

A literature review was conducted as part of the research activities [24] to understand how cameras and microphones were used, in combination with DL, to improve current railway maintenance activities.

Review Details. Despite the railway domain encompassing several “maintenance tasks”, the majority of the works based on audio and video data resulted to fall into two main categories:

- *Surface Defect Detection*, focusing only on surface defects (e.g., cracks in tunnels, rails head, etc.);
- *Defect Inspection*, including all models intended to detect and analyse different types of faults (e.g., broken objects, missing parts, anomalies in components, etc.).

A plethora of DNNs have been investigated in the literature to detect (obviously, visible and audible) defects within different “railway assets”, including those depicted in Figure 2.7.

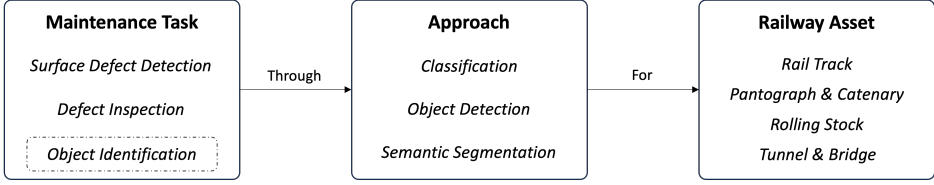


Figure 2.8. High-Level Picture of Approaches for Railway Assets Defect Detection through Video/Audio Data

Actually, in addition to the above *maintenance tasks*, a third category can be considered, encompassing all those promising works that did not directly address a specific maintenance task but proposed approaches that could support such practices; for example, approaches oriented at detecting small railway components.

Indeed, one of the main findings of the review concerned the fact that, in many cases, the components to be analysed were extremely small compared to the scenario depicted in the image. In these cases, directly applying an approach to detect defects would be quite challenging. Hence, some studies (e.g., [145, 158]) proposed multi-stage approaches oriented at first extracting the object from the image and then analysing its health status, while others (e.g., [85, 57]) mainly focused on *Object Identification* trying to propose solutions for the detection of small-scale railway components. The latter could also serve as a basis to support future maintenance applications [24]. Lastly, concerning the implemented “approaches”, they were subdivided according to some of the traditional tasks partially identified in Section 2.1.3, i.e.: *Classification*, *Object Detection*, and *Semantic Segmentation*. Figure 2.8 shows a high-level picture of the literature analysed in [24].

Instead of re-analysing all reviewed papers, in this section, attention is mainly posed on the findings and the directions that have been possible to excerpt from that review. For further details on reviewed studies, reference [24] can be easily visioned since it was published in open access.

Review Findings and Discussion. As for the majority of the approaches analysed in the previous Section 2.3.1, also in this case, they have mostly been leveraged State-of-the-Art DNNs, especially YOLO networks and Faster R-CNN which found wide application in almost all the possible

combination between the aforementioned railway assets and maintenance tasks. Particularly, they were adopted both to directly detect defects within components or as the first stage in multi-stage approaches (as introduced above) to extract railway components from the scene depicted in the image. Again, as in Section 2.3.1, most of the proposed approaches work quite well within the conditions set by the authors, however, it is not straightforward to understand which of them would be the most appropriate to address a specific maintenance task on a specific railway component. In [24], attempts were also performed to understand whether there were some publicly available datasets that could serve as reference benchmarks for research in those directions; where by “publicly available datasets” it was intended datasets published by some institution/entity or disclosed by papers’ authors as part of their work. Some well-known datasets were adopted, like ImageNet [27] or MS COCO [81], to pre-train DNNs (i.e., apply Transfer Learning, see Section 2.1.2) or to evaluate the performances of the implemented models against “general CV datasets”; however, only a few ad-hoc railway datasets were found [24]. Among those available exploited by researchers, AudioSet [64] resulted to be interesting. It is a collection of about 2 million audio clips and, rather than an ad-hoc railway reference dataset against which to test the performances of a DNN, it can be exploited to easily and timely build real-world audio datasets. It has been leveraged to collect part of the data used in Chapter 6.

Another aspect that is worth underlining concerns **where** these sensors were installed. Despite some tests conducted within laboratories, in the majority of cases, the sensors were part of special inspection vehicles (e.g., [137]) or were installed on rolling carts (e.g., [65]) (whether “manually pulled” or automatic); in other cases, instead, tests were performed by installing cameras on drones [17]. The main issue, in these cases, is that the shift from scheduled to continuous monitoring is barely supported; in addition, the cost-effectiveness of such sensors is reduced, given that they are installed on costly machines. Nonetheless, there has been evidence of DL approaches applied in combination with data captured from cameras mounted on the roof of operating trains (e.g., [80]). This would suggest that the vision of using (specifically) cameras to continuously monitor railway assets may be feasible. Then, in addition, as already mentioned in Section 2.3.1, cameras mounted on trains may be visionary adopted to both

perceive threats within the environment and acquire data for maintenance purposes.

To conclude, not many contributions were found leveraging audio data, which were mainly oriented at classifying, through CNNs, collected acoustic events representing rails or rail tracks defects (e.g., [77, 18, 143]).

2.4 Challenges for DL in Railways

From a theoretical perspective, as also witnessed by the reviews discussed in the previous section, Deep Learning, and Artificial Intelligence in general, could introduce several opportunities in railways to improve current practices or introduce new functionalities. However, despite the opportunities, there are some paramount challenges that should be overcome for the safe and fast take-up of AI/DL in railways.

Two surveys (questionnaires) were conducted by RAILS to collect opinions and concerns of *practitioners and researchers* worldwide to identify the main challenges to be overcome and the directions to be taken for the introduction of AI in railways. The first survey was conducted in 2020 [112] and was oriented, among other purposes, at identifying the *blocking factors* that were slowing the process of integration of AI in railways. Briefly, the top three identified challenges were: i) safety, dependability, and trustworthiness concerns; ii) lack of dataset for testing; iii) missing specific standards and regulations. Another survey was then proposed in 2023 [114], where participants were asked to identify the current state of development of specific AI applications and the main criticalities that should be faced when concretely introducing such applications in operational railway environments.

Taking into account the reviews performed in this chapter and some results from RAILS surveys and activities [138, 112, 114], the challenges that among others are slowing the process of integration of AI in railway are related to the *certification of AI systems* and *data availability*.

2.4.1 Certifiability of AI Systems

Safety-critical (railway) systems must be certified against strict safety standards to be deemed operational. AI-powered functionalities should

undergo the same process, however, AI introduces some concerns in terms of trustworthiness and dependability making its certifiability quite challenging [112, 114, 44].

The European Panorama

According to the High-Level Expert Group on AI (an independent expert group set up by the European Commission), the concept of *Trustworthy AI* encompasses three aspects that AI systems must meet [38]. They should be: i) **lawful**, i.e., they must respect laws and regulations; ii) **ethical**, i.e., they should be compliant with ethical principles and values; and iii) **robust**, i.e., from both technical and social perspective, it must be ensured that they are not harmful. In addition, in order to be deemed trustworthy, AI systems should be compliant with the *Ethic Principles* and *Key Requirements* [38] reported in Table 2.2.

These led, afterwards, to the definition of the “Assessment List for Trustworthy AI (ALTAI) for Self-Assessment” [37], which is a framework encompassing, for each *Key Requirement*, a series of more pragmatic criteria that AI systems should be compliant with. For the sake of knowledge, the European Commission published a technical report in 2021 to map the ALTAI criteria to autonomous vehicles [36]. Along the same line, reference [10] investigated the applicability of the ALTAI framework to Advanced Driver Assistance Systems in autonomous vehicles; the framework resulted to be applicable to these systems, but there seems to be still room for

Table 2.2. *Ethic Principles* and *Key Requirements* from [38]

Ethic Principles (EPs)		Key Requirements (KRs)	
<i>EP1:</i>	Respect for the Freedom and Autonomy of Human Beings	<i>KR1:</i>	Human Agency and Oversight
<i>EP2:</i>	Prevention of any Kind of Harm	<i>KR2:</i>	Technical Robustness and Safety
<i>EP3:</i>	Principle of Fairness	<i>KR3:</i>	Privacy and Data Governance
<i>EP4:</i>	Principle of Explainability	<i>KR4:</i>	Transparency
		<i>KR5:</i>	Diversity, Non-discrimination and Fairness
		<i>KR6:</i>	Societal and Environmental Well-being
		<i>KR7:</i>	Accountability

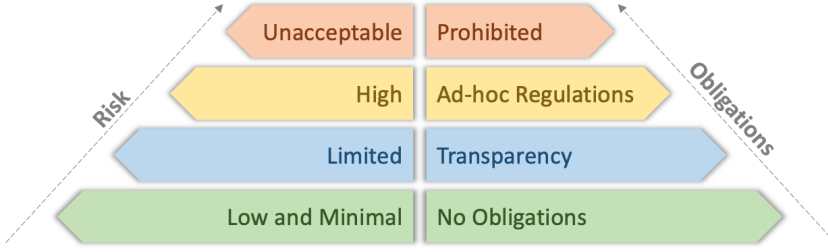


Figure 2.9. Risk-based Classification of AI Systems [90]

improvements to make ALTAI more suitable to these systems, including ALTAI variants specifically designed for each development phase.

In 2021, the European Commission also drafted a proposal for the introduction of the “AI Act” [35], a regulatory framework intended at horizontally regularising AI systems. Conceptually, it focuses on classifying AI systems according to a *risk-based approach* and providing specific requirements and obligations for each of the identified risk classes [90] (shown, for the sake of completeness, in Figure 2.9). Basically:

- AI systems presenting *Unacceptable Risks* for “people’s safety, livelihood, and rights” (e.g., social scoring) are completely banned.
- AI systems catalogued as *High Risk* are those that could impact people’s safety or rights. These would be subject to ad-hoc regulations covering various aspects such as data governance, record-keeping, transparency, human oversight, and so on. For the sake of knowledge, by consulting Annex III of the proposal [35], it is possible to find the following wording among *High Risk* AI systems:

“Management and operation of critical infrastructure: AI systems intended to be used as safety components in the management and operation of road traffic and the supply of water, gas, heating and electricity”.

It would not be so unlikely to have safety-critical railway systems in this area as well, or, at least, under the macro category of *High Risk* AI systems.

- AI systems classified as *Limited Risk*, e.g., chatbots or applications

for image/video generation, would be required to be compliant with only transparency obligations.

- Lastly, AI systems presenting *Low or Minimal Risks* would not have to meet any strict requirement.

Technical Issues for DL

Among the various *Ethical Principles* and *Key Requirements* that should be considered while designing Trustworthy AI systems, those related to their explainability and transparency are of particular interest when talking about DL applications.

Most AI approaches, especially DL ones, historically suffer from two major technical issues which make them poorly predisposed to be evaluated and certified through *traditional* safety standards [44]. First, most DL models are **opaque** (typically referred to as black boxes), i.e., the reasoning the model performs to map a given input with a specific output may not always be understandable by humans. Second, DL models may be **unstable**, i.e., to slight variations of the input could correspond heavy variations on the output [92].

As for the *opacity*, this problem affects the fourth Ethic Principle and most of *Key Requirements*, especially the fourth also in this case (see Table 2.2). Different approaches, falling under the name of *Explainable AI* [5], have been proposed in the literature with the intent of unwrapping black-box DL models and, indeed, provide explanations about their functioning. Despite these techniques could be extremely helpful to increase the level of transparency of DL models, they are not free from shortcomings. First, transforming opaque models into comprehensible and readable ones would not only benefit railway operators but also external attackers who would have more information about the implemented model; it would be much easier to extract sensitive data from readable models. Second, *Explainable AI* approaches may be targeted by adversarial attacks [47]; which would compromise the trustworthiness of the explanations themselves. Third, other complications arise with multi-modular systems, where more than one module would leverage DL; in this case, each module would require to be explained, explanations should benefit different operators, from developers to regulatory bodies and end-users, hence, each module would require

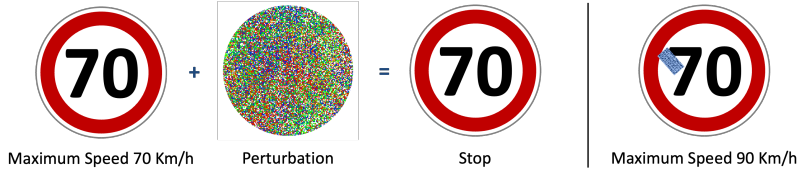


Figure 2.10. Exemplification of Causes that could Fool a DNN

different explanations at different levels of abstractions [102].

Concerning *instability*, this issue mainly affects the second Key Requirement, i.e., the robustness and safety of AI systems. This is because adversarial attacks [121] may take advantage of unstable systems and mislead them to produce wrong outputs. Figure 2.10 exemplifies a couple of scenarios that could occur on image data; the image shows a traffic sign recognition problem, however, the same problems could generally apply to other recognition tasks. Basically, adversarial attacks may over-impose some specific patterns (which may also be imperceptible to the human eye) on input images to induce DL models to produce an incorrect classification; moreover, DL models can be also fooled by physical patches or even shadows [56].

What has been discussed so far represents only a subset of the problems that could affect DL models when it comes to assessing their trustworthiness or, visionary, SIL. Some expedients, e.g., the application of the *safety envelope* to *train driving systems* [44], could be adopted to mitigate the effects of these issues, however, well-defined procedures and standards would be required to effectively validate and certify AI systems.

The ATP as the Safety Envelope for Intelligent Train Operations

The safety envelope architectural pattern [74] could be exploited to progressively introduce AI in train driving systems starting from non-safety-critical functionalities [44]. The idea is to have a system acting as a “checker” that defines a *safe operational state space* (free from any hazard) within which the train is free to move in autonomy. Then, any action that would bring the whole system outside the safe state should be intercepted by the checker which would apply proper protection mechanisms [44, 116]. This is, from a high-level perspective, what already happens between ATO and

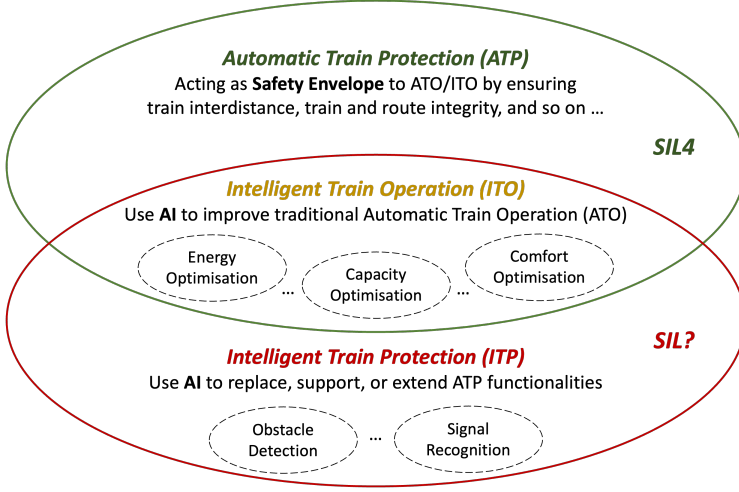


Figure 2.11. Introducing AI in Autonomous Trains [44]

ATP. Their interconnection can be declined in terms of safety envelope as follows [116]: the **ATP** acts as the checker and ensures that the correct dynamic speed profile is applied (to avoid derailments, collisions, and so on); the **ATO**, instead, acts as the “doer”, and performs the actions required to automatically move the train from one station to another. The **ATP** constantly monitors the actions taken by the **ATO** and, in case of anomalies (e.g., the **ATO** accelerates over the maximum admitted speed) the **ATP** would apply specific protection mechanisms to avoid unpleasant consequences (e.g., by activating emergency brakes) [116].

In the same way, the **ATP**, which can be rated *SIL*₄, could act as the checker for an AI-empowered **ATO** system, i.e., *Intelligent Train Operation (ITO)* [44]. The **ITO** could then visionarily implement AI/DL functionalities oriented at optimising energy or improving line capacity, while the **ATP** would take care of all safety aspects. Conversely, when wanting to apply AI to empower **ATP**, or to build complementary systems managing safety aspects that the **ATP** cannot handle, like obstacle detection, those intelligent systems should be somehow certified by taking into account ad-hoc standards and procedures. Figure 2.11 graphically summarises this scenario. These aspects will be recalled later in Chapter 3.

Towards Certifiable AI Systems

Standards and practical guidelines for AI systems helping in their certifiability process have been under the spotlight over the recent years in different domains, including transport. In avionics, for example, the European Union Aviation Safety Agency proposed a W-shaped process to assess the usage of neural networks by considering a visual traffic detection system as a case study [33]. In addition, they also published some preliminary guidelines for the adoption of ML applications oriented at assisting human operators (Level 1 ML) or cooperating with them (Level 2 ML) [31, 32]. As for standards, some of them appear to have been approved in the United States for AI-powered autonomous roads vehicles (e.g., ANSI/UL 4600 [142]) [50]. These proposals and standards are clearly oriented at other sectors, however, they could be used as a starting point also in railways. Preliminary investigations have been performed in [108, 50], where the authors investigated the applicability of ANSI/UL 4600 to ATs; to be precise, to AI-powered OD systems in freight trains [50].

Furthermore, as for standards addressing AI concepts in a cross-sector manner, it is important to mention the CEN-CENELEC Joint Technical Committee 21 “Artificial Intelligence”⁷, which, starting from other already existing standards, focuses on developing and adopting standards for AI and related data, and providing guidelines to other committees concerned with AI (e.g., by supporting the upcoming “AI Act”). Along about the same line, the IEEE Standards Association proposed a “Global Initiative on Ethics of Autonomous and Intelligent Systems”⁸ whose principal goal is to make sure that autonomous and intelligent systems would benefit humanity. In this context, a landmark paper (“Ethically Aligned Design” [140]) was issued with the aim of providing guidelines and recommendations for the development of regulations, certification, and standards for the design of Autonomous and Intelligent Systems. This initiative also encompasses a series of standards (IEEE 7000) that are being developed with the intent of providing procedures and guidelines for the implementation of intelligent systems and related concerns (e.g., data privacy and ethics).

To conclude with standards, also explainability and Explainable AI

⁷<https://www.cencenelec.eu/areas-of-work/cen-cenelec-topics/artificial-intelligence>

⁸<https://standards.ieee.org/industry-connections/ec/autonomous-systems>

approaches have been under the spotlight so far, being important aspects for trustworthy AI systems. The IEEE Standard Association eXplainable AI Working Group⁹ is working on a Standard for Explainable AI¹⁰ whose expected submission date is around mid 2024.

2.4.2 Data Issues: Quality and Availability

Besides the certification problems analysed above, there are other issues that are challenging researchers and practitioners approaching the development of DL solutions for railway applications.

As identified within both the reviews in Section 2.3, most of the approaches proposed in the literature performed efficiently under the conditions set by their authors, including the dataset. The majority of researchers built ad-hoc datasets (from scratch) to perform their analyses; where, “build” represents the combination between *data collection* and *data labelling* (if required by the approach). These two phases are of paramount importance when it comes to training DNNs as data, in the form of weights (see Section 2.1.2), would be integral part of the trained network. A trained DNN is capable of solving a specific task based on the knowledge extracted from the dataset; if the dataset does not properly represent the real task that is intended to be solved, the network might address a problem related to a distorted version of reality. There is actually another scenario: the dataset represents only part of the real task. A trivial example is given by the number and types of objects considered in a dataset built ad-hoc for an obstacle detection problem: established that it would be nearly impossible to build a dataset containing all possible types of objects, the problem would be in the rationale adopted to select only a subset of obstacles. The same would hold for datasets taught to address some defect detection problems that could represent only a portion of the possible defects that must be detected for proper maintenance.

Premised that a problem could be subdivided into multiple tasks and hence different DL models, each of which addresses a sub-task, could be merged together (e.g., Model A recognise cars and pedestrians, while Model B recognise animals), what seems to be missing are *ad-hoc guidelines* for

⁹<https://sagroups.ieee.org/2976>

¹⁰<https://standards.ieee.org/ieee/2976/10522>

the generation of qualitatively optimal datasets for railway applications that would support researchers in data collection, or even *standardised datasets* (discussed below) to be used as *benchmarks* to train and test DL applications [24, 138, 112, 114].

Data Quality

The problem of having a qualitatively optimal dataset is extremely sensitive in railways, especially when dealing with safety applications. The process towards high-quality datasets is not straightforward as, visionary, it would require the cooperation of AI experts, railway stakeholders, and regulatory bodies who have great expertise on the task to be solved [118].

A series of guidelines, composing what can be defined as a “standardised process”, should be identified for each sensitive task. These would include, among others, the specific sensors to be used, how data collection should be performed, what are the main characteristics of the problem that must be taken into account, which type and what events should be collected, and so on [24, 118]. Clearly, being guidelines, they could not precisely define each single aspect (this would be preferable), rather, they could specify ranges within which researchers should move to obtain a qualitatively optimal dataset for a specific task. These *standardised processes* would contribute to homogenising the datasets collected by researchers, potentially making the solutions they propose easily comparable.

Strictly linked to *standardised processes*, “standardised datasets” would widely benefit the railway research community. Conceptually, a *standardised datasets* is a dataset collected by following a specific *standardised processes* which is made available by some reliable entity and that can be used as a *benchmark* to test and train AI/DL applications [24, 118]. They can also be used to promote competitions to support the technology advancement in railways, similar to those built around publicly available datasets like ImageNet and MS COCO which have largely contributed to the development of State-of-the-Art DL models for CV applications.

Data Availability

The *data availability* problem may principally have two connotations [26]. First, datasets collected in the context of scientific works may not be

disclosed by the authors; as a side effect, their study cannot be properly reproduced. Second, the events that should be collected are rare by nature (e.g., failures in safety-critical systems) or potentially harmful (e.g., obstacles on rail tracks), therefore, it would be extremely time-consuming or unsafe to collect them in operational environments. Actually, there would also be a third one: data have nowadays become extremely valuable assets for companies, including railway enterprises, therefore, it is reasonable that they tend to protect their data instead of disclosing them [114].

When it comes to training AI/DL architectures, *real data*, i.e., collected in real environments, would be preferable since they would represent the actual reality the system should face at run-time. Nonetheless, not everyone has the possibility to access them either because these data exist but have not been made available, or because these data do not exist and it is not possible to access real systems to collect them. *Standardised datasets* would play a key role in this context and, potentially, would be the best choice. Alternatively, there exist some datasets that can be adopted to investigate DL approaches for some specific tasks [105, 24]. Examples in the context of vision-based OD are RailSem19 [165], which has already been used within some scientific works (e.g., [12, 11]), and the recently disclosed OSDaR23 [134]. However, these datasets may not specifically suit (or suit only in part) the problem that is intended to be addressed.

Two main scenarios can be identified [26]: i) *only limited amount of data has been collected/is available* and ii) *no real data are available*.

In the first case, “limited” means that the amount of data is not enough to allow a proper characterisation of a DL model. There exist some techniques that can be exploited in this context:

- *Data Augmentation*. These techniques mainly aim at producing new synthetic samples to increment the quantity and diversity of collected data [157]. By considering an image dataset as a practical example, basic transformations like cropping, rotation, and brightness increment can be applied to increase the number of samples. In addition, some advanced solutions based on DL (e.g., GAN [2]) can be adopted. A specific example of advanced methods for data augmentation for vision-based OD is given in [45], where a framework based also on DL is adopted to properly introduce (e.g., by respecting the perspective) obstacles in real images depicting free rail tracks.

- *Transfer Learning*. Already discussed in Section 2.1.2. Worth mentioning, existing datasets, such as the aforementioned RailSem19, can be adopted for this purpose to pre-training the network that would then be fine-tuned on the collected/available data. This is what has been done in this Thesis, further details are given in Chapter 5 and Chapter 6.
- *Federated Learning* [79, 156]. This may be of primary importance in model training rather than data sharing or data collection. Basically, a model can be trained through independent sessions on different data; then, for example, multiple companies could contribute to the training of the same DL model without having to directly share their (confidential or valuable) data.

As for the second scenario, i.e., when no data are available, it would be possible to exploit, among others, the following solutions:

- *3D Editors and Real Life Simulators*. These can be exploited to generate virtual scenes from which to extract video, but also other types of data. Examples of 3D editors are, among others: RoadRunner, a MathWorks editor conceived to generate scenarios for autonomous driving but easily adaptable to railway scenes; and Unreal Engine, conceived for the realisation of video games which can also be exploited to generate virtual railway scenarios and automatically extract (and also label) data [30]. Then, for Real-Life Simulators, they are intended actual video games, like Grand Theft Auto (GTA) V, which can be exploited to collect synthetic data (e.g., [93]). All the aforementioned software have been used to generate synthetic data to train and test most of the DNNs adopted in this Thesis; further details will be given in Chapters 5 and 6.
- *Digital Twins*. This could be confusing as, to build a DT, the real asset should be equipped with adequate (IoT) sensors, hence, data should be available. However, this would be specifically the case when the problem to be addressed regards the analysis of failure/faults of the system under examination, especially if those events are rare. There could be adopted some injection approaches [103] to stress the DT and possibly generate synthetic anomaly data.

- *Generic Data Collections and Multimedia Platforms.* There are some “generic” datasets or collections containing a lot of data, but are not tailored to railway tasks (differently from datasets like RailSem19 that are born within railways). Examples may be ImageNet, MS COCO, and AudioSet. These can be adopted either to extract some data to build some ad-hoc datasets, perhaps enriched with new data collected from scratch, or to pre-train a DNN (to apply TL). Continuing, in some cases, these are related to well-known challenges and can also be adopted to have a reference about how the implemented model performs compared to others in the literature on a generic task, e.g., the detection of about 80 objects that MS COCO contains. To conclude, multimedia data are nowadays widely shared online, and there are some specific platforms that could be exploited to easily and timely retrieve data (e.g., YouTube¹¹). AudioSet and YouTube have been exploited in the context of this Thesis, further details are given in Chapter 6.

To conclude, as mentioned at the beginning of this section, both *data collection* and *data labelling* are of paramount importance. Once collected, data should be properly labelled to match the task to be addressed. Usually, this is done manually, resulting to be extremely time-consuming and, sometimes, imprecise. In the case of synthetic data generated through 3D Editors, some of them, like Unreal Engine, could integrate functionalities that can be exploited to automatically label data while extracting them [30]. As for real data, or even synthetic one that cannot be labelled automatically, they exist some Labelling tools that facilitate and support the labelling process^{12,13,14}; alternatively, semi-automatic approaches based, e.g., on self-training (like in [167]), can be implemented to label a large dataset starting from a small subset of labelled data. In this Thesis, they have been implemented an automatic labelling through Unreal Engine and a semi-automatic approach based on self-training and transfer learning to label synthetic data, extracted from Unreal Engine and RoadRunner respectively, which will be discussed in Chapter 5.

¹¹<https://www.youtube.com>

¹²<https://segment-anything.com>

¹³<https://keymakr.com/annotation-tool.php>

¹⁴<https://www.v7labs.com>

Introducing DL in Intelligent Train Control

The definition of AI given in Section 2.1 allows for a clear classification of the train driving systems typically adopted in railways. In Automatic Train Operation (ATO) systems (see Table 2.1) currently adopted within GoA4 metro lines – allowing trains to run without drivers or crew members – the autonomy with respect to train control strategies is almost null as safety-critical decisions are centralised in a Zone Controller [44]. For the sake of clarity:

- *Automation* (Automatic) refers to the capability of a system to, indeed, *automatically* or *semi-automatically* perform some actions considering some pre-specified roles.
- *Autonomy* (Autonomous) refers to the capability of a system to dynamically adapt to changes in the operational scenario by taking independent decisions by leveraging procedures on methods that could also involve AI [44].

Currently, most operational driverless trains are basically *automatic* rather than *autonomous*.

As mentioned in Section 2.2.1, while in *segregated* railways it has already been reached a high level of *automation*, in *open* railway environments

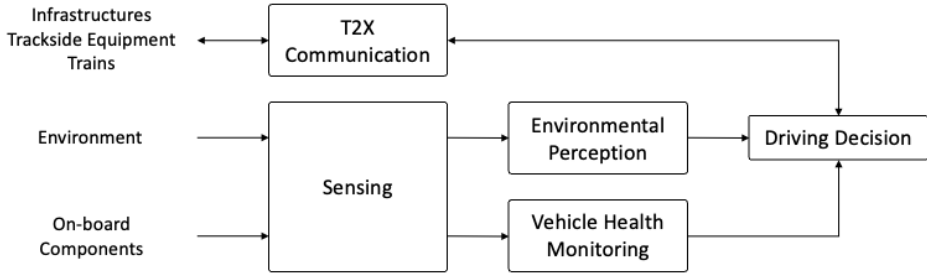


Figure 3.1. An Example Architecture for Autonomous Trains

research towards driverless trains is still ongoing given the large interoperability requirements posed by these complex scenarios and potential threats that would not be possible to manage through pure automatism [44].

This chapter provides a *vision* – i.e., a series of considerations and proposals – to move towards AI-empowered Autonomous Trains (ATs), with a specific focus on *open* railways. Hence, they are defined a **conceptual shift** and some **structural needs** that would be required to support the step-wise integration of DL (but also AI) functionalities in train driving systems.

The ideas and recommendations discussed below are not intended to be exhaustive and, clearly, more in-depth analyses would be required to assess their actual implementability. This *vision* has then the objective of providing a *high-level roadmap* for the introduction of AI in AT driving systems, whose feasibility should be supported, in the future, by further investigations and proper evaluations.

3.1 An Example Architecture for Autonomous Trains

As already mentioned, one of the main challenges that *open* railway environments pose concerns the management of particular threats that cannot be properly mitigated through only formalisms or automatism (e.g., intrusions on rail tracks). This translates into the need to equip trains with special mechanisms that allow them to analyse the surroundings, but also the health status of their internal components, in order to have **full**

situational awareness to properly identify the best action to be taken to safely proceed with the journey.

On this basis, and inspired by some studies (e.g., [153, 28]), Figure 3.1 proposes an example architecture for ATs identifying some of the main components, yet not all of them, that an AT should implement to achieve what stated above by:

- Receiving information from infrastructures or other agents, e.g., other trains on the same line (*Train to Everything (T2X) Communication* module).
- Perceiving information from on-board components and the environment (*Sensing* module).
- Constantly monitoring the health status of its onboard components (*Vehicle Health Monitoring* module).
- Constantly monitoring the environment to recognise possible threats preventing safe operations (*Environmental Perception* module).
- Taking decisions in autonomy based on all information acquired (*Driving Decision* module).

Then, the first four modules would allow for what has been previously mentioned as *full situation awareness*, i.e., they give the AT all the necessary information to understand what is happening onboard and in the surroundings.

What just described is clearly a high-level view of the modules that an AT should include to possibly ensure safety in operational environments. However, as explained in Section 2.4.1, the most blocking factor in implementing such architecture is the lack of proper standards against which to estimate the SIL (or equivalent measure) of AI-empowered systems.

3.2 Conceptual Shift: Grades of Intelligence

A possible approach would be to gradually introduce AI starting from non-safety-critical functionalities. *What* would be needed in this context is

a new classification for railway lines mapping the adoption of AI functionalities in train driving systems. To that aim, some Grades of Intelligence (GoIs) are introduced, which would drive the step-wise integration of AI [44]. Important to mention, these GoIs build upon currently adopted GoAs (Section 2.2.1) and are not intended to replace them. These two “classifications” evaluate different aspects of the same railway line; GoIs should be intended as an extension of GoAs, allowing for the identification of the, indeed, grade of intelligence implemented in a railway line which would not be possible if only GoAs are taken into account.

As GoAs reflect the presence or absence of Automatic Train Systems (see Table 2.1), GoIs reflect the presence or absence of *Intelligent Train Systems*. The latter can be seen as AI-empowered versions of the former, e.g.: the Intelligent Train Operation (ITO) improves and/or extends the functionalities of the traditional ATO by introducing intelligent behaviours in order to optimise energy consumption, line capacity, and so on; similarly, the Intelligent Train Protection (ITP) improves and/or extends the functionalities of the traditional Automatic Train Protection (ATP) by implementing functionalities like signal recognition or obstacle detection. Important to highlight, according to this *vision*, **Intelligent Train Systems are NOT intended to replace** Automatic ones, rather, they would act as complementary or supportive systems.

Figure 3.2 graphically represent the various GoIs which are summarised herein [114]:

- **GoI1 - Limited/No Autonomy:** AI is not used in any safety-critical functionality but can be adopted to enhance or support Automatic Train Supervision (e.g., optimise train scheduling).
- **GoI2 - Partial Autonomy:** Intelligent Train Operation (GoI2.1) **OR** Intelligent Train Protection (GoI2.2) are implemented. GoI2.1 and GoI2.2 are *mutually exclusive* and represent two different ways through which partial autonomy can be obtained:
 - **GoI2.1: ITO** is implemented to improve/support ATO. For the fast take up of AI in train operations, the concept of *safety envelope* introduced in Section 2.4.1 could be exploited. In this scenario, “*traditional*” protection systems (such as the ATP or

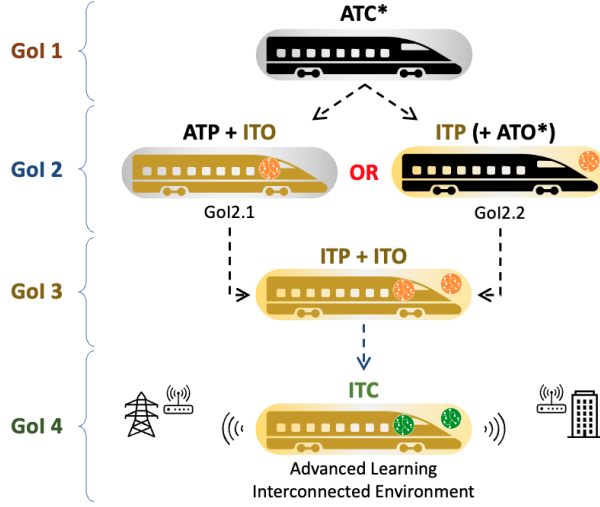


Figure 3.2. Grade of Intelligences
(* where/if present)

equivalent) would be *mandatory* to monitor **ITO** actions and provide protection when required.

- **GoI2.2:** **ITO** is implemented to improve, support, or (potentially) deputise **ATP**. In case of **ATP**, **ITP** would act as a complement systems to manage events not handled by the **ATP** (e.g., obstacle detection). Conversely, in case the **ATP** is unavailable, e.g., when it is not present at all or when it fails obliging to partial supervision, the **ITP** could deputise the **ATP** by, e.g., automatically recognising signals and/or obstacles.

GoI2.1 would potentially be easier to achieve than **GoI2.2**, since, in the case of **ITP**, **AI** would **directly manage safety aspects**; therefore, to be considered operational, the **ITP** would have to be certified as **SIL4**. Nonetheless, this goal seems not to be achievable at the moment given the current panorama on ad-hoc **AI** safety standards and all the concerns introduced in Section 2.4.1.

- **GoI3 - Full Autonomy:** both Intelligent Train Operation and Intelligent Train Protection are in place to allow trains to operate

unattended. This level, however, does not expect any advanced learning functionalities, e.g., the possible vision-based obstacle detection implemented through DL by the ITP is trained beforehand and not updated in real-time.

- **GoI4 - Full Autonomy in Fully Connected Environments:** like GoI3, GoI4 expects both Intelligent Train Operation and Intelligent Train Protection. Nonetheless, in this case, they would also be implemented:
 - Advanced, online learning capabilities to let AI models to constantly evolve over time to understand how to further optimise operations.
 - A fully-connected ecosystem, dynamically updated. ATs would then be constantly fed with information concerning safety and operability aspects coming from infrastructures or any other agent or asset (e.g., other trains, level crossings, and stations).
 - Advanced analysis mechanisms giving trains the capability of constantly monitoring the health status of on-board components.

The latter aspect could also be implemented at lower GoIs. It is explicitly mentioned here because it would be specifically needed at GoI4 as, together with ITP, it would benefit from the interconnected environment (as explained below) and provide trains with all relevant information to achieve *full situation awareness* and “consciously” take safe actions.

3.3 Structural Needs: Implementing Grades of Intelligence

GoIs would help to classify railway lines according to the type of AI functionalities implemented (e.g., train driving operations, protection, and so on). In order to achieve GoI3/4, however, according to the *vision* subject of this chapter, it would be required to arrange AI systems according to a specific structure that would define their *field of action* and *where* they should be implemented. This could be formalised by introducing some *Levels of Intelligence (LoIs)* [44, 114]:

- **Edge Intelligence Level (Edge Level).** Intelligent systems mounted on-board railway assets (e.g., trains) would implement *local autonomy* (e.g., on-board obstacle detection). Such systems may have lower computing power compared to systems deployed at the other LoIs; nonetheless, they would benefit from shorter communication links providing them with reduced response times.
- **Fog Intelligence Level (Fog Level).** At this Level, intelligent systems would manage multiple assets handling their relationships to achieve comprehensive optimisation. An example is given by capacity optimisation through the implementation of functionalities like Virtual Coupling which would be orchestrated and managed by the systems deployed at the Fog Level based on a larger knowledge of what is happening within an entire railway line.
- **Cloud Intelligence Level (Cloud Level).** Intelligent systems withing this Level would be in charge of analysing large amount of data (collected, for example, from multiple assets installed globally) to gather or extract knowledge and information that could be shared with lower LoIs or exploited to implement more comprehensive systems. A failure prediction system, for instance, can be implemented by leveraging data from multiple installations of the same asset as these would most likely provide more information about possible malfunctions than data gathered by monitoring a single asset only.

Such LoIs would allow for the separation of concerns between AI systems at different Levels in order to structure a multi-level architecture for GoI4 railway lines which, in the context of this Thesis, is exemplified in Figure 3.3.

Important to emphasise, this architecture is not taught to be exhaustive: the identified modules are those that would be significant in the context of this *vision*, but others may be required to properly allow trains to safely behave autonomously. An example is given by *traditional* modules and systems, like ATO and ATP, which are not indicated in the figure, but would be of primary importance to implement some specific functionalities (e.g., the Safety Envelope discussed in Section 2.4.1 and above).

To conclude with this preamble, the example architecture in Figure 3.3 shows the modules that would be required for GoI4 railway lines according

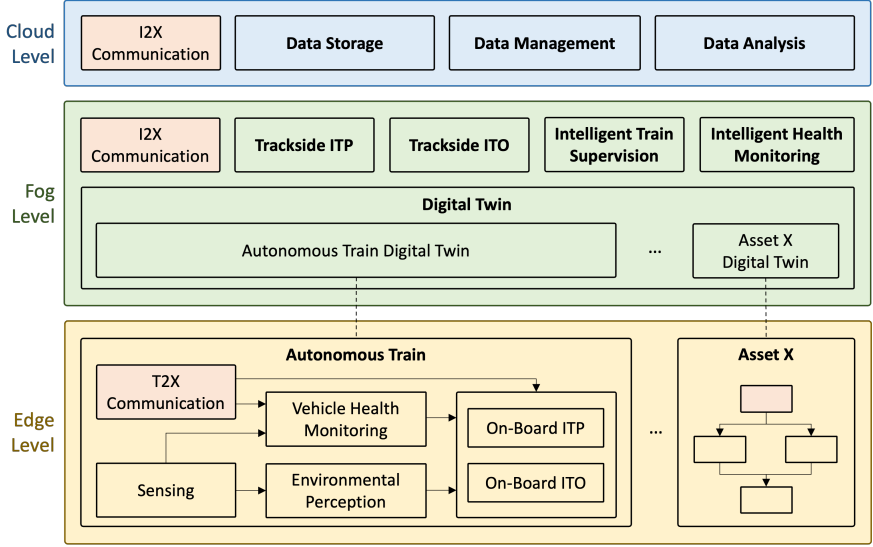


Figure 3.3. An Example Architecture Towards GoI4

to this vision; all other GoIs would be achieved by removing some of them as explained in the following section.

The example architecture in Figure 3.3 encompasses the following Levels and modules:

- **Communication Modules.** *Communication* is transversal to the LoIs. *Communication modules* would be expected at each LoI (to allow, for example, Infrastructure to Everything (I2X) communication) and for each asset within the Edge Level (to allow, e.g, Train to Everything (T2X) communication).

For the deployment of AI-powered autonomous driving functionalities, communication across assets and LoIs would be essential. For instance, in order to establish a proper driving policy, Virtual Coupling (as analysed in [115, 116]) necessitates that trains constantly and reliably communicate with trackside infrastructures, systems deployed at the Fog Level, and other trains or assets; in this scenario, with trains running closer together, disruptions or delays in communication could lead to unfavourable consequences.

- **Cloud Intelligence.** This Level would include functionalities oriented at storing, managing, and analysing data from multiple instances operating (potentially) globally:
 - **I2X Communication.** The I2X Communication module would allow the exchange of information with lower LoIs. Whether the knowledge elaborated at the Cloud Level could be directly passed to the Edge Level (bypassing the Fog Level) or should be first managed by the Fog Level is beyond the scope of this analysis; that should be deeply evaluated in the future considering related opportunities and shortcomings.

For example, within the *Data Analysis* module (presented below), an AI system oriented at detecting malfunctions within a specific railway component may be optimised leveraging data coming from multiple instances of the same component installed on different assets. Hence, the system in charge of monitoring a specific instance (i.e., a specific railway component) at the Edge Level should be updated with the “new knowledge”. The path that this knowledge should travel to reach the Edge Level system is something that should be evaluated in the future.
 - **Data Storage.** This module would store data and information received from the *Data Management*.
 - **Data Management.** The main responsibilities of this module would include the pre-processing of data (if necessary) and the sorting of data between *Data Storage*, *Data Analysis*, and lower LoIs.
 - **Data Analysis.** This represents the core module of the Cloud Level. Essentially, it would incorporate all the necessary functionalities to train, test, and update AI/DL models. At this Level, as already mentioned, AI models would be trained with data coming from hundreds of instances installed globally stored by, indeed, the *Data Storage*. Models trained in this manner would probably be more efficient, as they would have more information from which to extract knowledge than that retrievable from a single instance (e.g., at the Edge Level).
-

- **Fog Intelligence.** This Level would manage all the assets and systems operating in a specific railway sub-network; this would translate into multiple parallel Fog Levels under the same Cloud Level. The modules expected for this Level are:
 - **I2X Communication.** As for the Cloud Level, this module would allow the Fog Level to exchange information with the Cloud and the Edge Levels.
 - **Trackside ITP.** This module would use AI-empowered functionalities to manage protection leveraging data from Cloud, Fog, and Edge Levels (including “traditional” protection systems like the ATP).
For instance, the *Trackside ITP* would determine whether the train’s speed must be adjusted in order to guarantee safety when the train is approaching a potentially defective LC.
 - **Trackside ITO.** The ITO at the Fog Level, knowing the status of the various assets in a specific sub-network, would handle and sort the necessary information to allow all trains in that sub-network to proceed autonomously. For instance, to enable Virtual Coupling, *Trackside ITO* could calculate in real-time the ideal trajectory that every train should take; then, each train’s *On-Board ITO* (explained below) receives the computed trajectory and actuates the proper driving policy.
 - **Intelligent Train Supervision.** To effectively optimise (or maximise) railway line utilisation and throughput, *Intelligent Train Supervision* would extend ATS functionalities and offer suitable train routing solutions (e.g., promptly responding to disruptions, optimising scheduling, and so on) by exploiting AI.
 - **Infrastructure Health Monitoring.** This module would include and manage all functionalities oriented at performing predictive maintenance. It would monitor all the assets deployed within the sub-network to predict or detect possible malfunctions that could compromise safe operability.
 - **Digital Twin.** Ideally, to each asset in a sub-network would correspond a DT; these would be managed and coordinated
-

by this module. **DTs** could be leveraged to analyse the status of the assets at the Edge Level to optimise operations and increase the safety. For instance, if **DTs** for both trains and **LCs** can be generated, these can be adopted at the Fog Level to promptly apply protection mechanisms in the event that the train is approaching a faulty **LC**. Further details are provided in [116].

- **Edge Intelligence.** The Edge Level represents an ensemble of assets (from *Autonomous Trains* to other generic railway assets exemplified by *Asset X* in the figure), whose functionalities, or part of them, are improved by means of **AI**. The identification of all possible assets and related modules/functionalities is beyond the scope of this Thesis; here, the attention is mainly posed on the modules (some of which have been already mentioned when discussing Figure 3.1) that would be essential for an **AT** at **GoI4**:
 - **T2X Communication.** This module would allow an **AT** to send/receive information to/from other assets and upper Levels.
 - **Sensing.** This module would manage all the sensors installed on-board, whether oriented at monitoring the environment (e.g., cameras and LiDARs) or on-board components, and apply pre-processing when needed.
 - **Vehicle Health Monitoring.** This module would implement functionalities allowing the **AT** to self-analyse the health status of on-board systems/components (e.g., failure detection/prediction and anomaly detection).
 - **Environmental Perception.** This module would implement the adequate mechanisms giving the **AT** the capability of autonomously analysing the surrounding environment (e.g., obstacle detection, signal recognition, and so on).
 - **On-Board ITO.** Leveraging data from other **AT** modules (especially the *Communication* one which connects the **AT** with other assets and upper Levels), this module would decide the optimal driving policy to adopt.
-

- **On-Board ITP**. This module would monitor the activities of the *On-Board ITO* and, exploiting data from the other module (i.e., *Vehicle Health Monitoring*, *Environmental Perception*, and *Fog* modules), apply protection mechanisms if necessary (e.g., emergency brakes in case an obstacle is approaching, adjust speed, and so on).

3.4 From Limited/No Autonomy to Full Autonomy

All modules in Figure 3.3 would be required to allow trains to receive all the necessary information to safely run autonomously. However, not all the indicated modules are required at GoIs lower than GoI4. As already mentioned, *Automatic Train Systems* (Table 2.1) *are not bypassed*: at the current stage of development of AI systems, for example, ATP would be essential at GoI2.1 to handle possible ITO failures. Automatic Train Systems, especially ATP or equivalent, should be considered almost mandatory at every GoI, at least until the ITP technology would be mature enough to be adopted to improve safety also in railway lines which are not equipped with any ATP-like system.

Having stated that, the steps to follow towards the achievement of GoI4, also taking into account the example architecture in Figure 3.3, are outlined below.

Step 1: Towards GoI1

Conceptually, current railway network are already GoI1. Indeed, this grade has been conceived as connection ring between what has already been implemented in railways and what could be done with AI. As mentioned, GoIs are detached from GoAs and, at GoI1, it would be possible to have from GoA0-GoI1 lines up to GoA4-GoI1 lines. The latter case represents the situation where trains are capable of operating unattended, but no AI is involved. Then, with respect to the architecture in Figure 3.3, none of those AI-empowered modules would be strictly required, but for the *Communication* modules which are crucial at any GoI to exchange any sort of information. In addition to that, AI could be adopted to improve

or support non-safety-critical practices which could be double-checked by human operators.

Step 2: From GoI1 to GoI2

The shift from GoI1 to GoI2 can be performed in two ways: *towards GoI2.1* or *towards GoI2.2*.

Step 2.1: Towards GoI2.1. At GoI2.1, Intelligent Train Operation is implemented both as *Trackside ITO* and *On-Board ITO*; however, ITP is not admitted. All functionalities that would allow ITO to operate correctly should be in place. The ATP will be mandatory, as it should ensure safe operability by protecting the overall railway system from ITO failures (i.e., it should act as the Safety Envelop for ITO). For instance, the ITO could implement DL-based strategies to compute the optimal train speed to save energy; if the computed speed exceeds the admitted one computed by the ATP, this should take the proper protective action(s) to ensure safety.

Step 2.2: Towards GoI2.2. At GoI2.2, Intelligent Train Protection is implemented at both Fog and Edge Levels; nonetheless, ITO is absent. All modules supporting and allowing ITP are required, for example, the *Environmental Perception* module in ATs would allow for the implementation of functionalities oriented at monitoring the environment looking for potential hazards (e.g., obstacle detection) which would be crucial for *On-Board ITP*. The main blocking factor, in this case, is represented by the certifiability of AI systems, which should be rated SIL4 before being introduced in operational environments.

Step 3: From GoI2 to GoI3

GoI3 “fuses” together the aspects in GoI2.1 and GoI2.2. Both ITP and ITO are implemented, together with all the functionalities that would make their interaction possible. Essentially, about all modules in Figure 3.3 would be necessary, with a very few exceptions: the *Data analysis* module at the Cloud Level, the *Health Monitoring* modules at both the Fog and Edge Levels, and the *Digital Twin* module. These could be present but are not mandatory according to this *vision*; indeed, these have been taught to allow for extremely advanced functionalities (e.g., full situation awareness of the whole railway line) which would be part of GoI4. However, the

combination between ITO and ITP would open for valuable opportunities like Virtual Coupling [115, 116], which would take full advantage of all the communication, analysis, and perception modules at any Level of Intelligence (LoI). Clearly, before moving towards GoI3, it would be essential to solve all certifiability issues related to AI systems.

Step 4: From GoI3 to GoI4

As the last step, the migration from GoI3 to GoI4 would have as the main objectives the digitalisation of railway assets, the setup of reliably connected infrastructures, and the possibility of implementing advanced AI systems capable of constantly updating their knowledge. Therefore, GoI4 requires the presence of all the modules allowing for advanced knowledge, advanced data analysis, and full situation awareness along the whole, at least, sub-network. Essentially, this translates in having all modules in Figure 3.3, including health monitoring systems and DTs. Actually, as already mentioned, *Data Analysis*, *Vehicle Health Monitoring*, and *Infrastructure Health Monitoring*, would not be required at GoIs lower than 4, however, they could be implemented to improve data analysis and situation awareness.

In conclusion, some high-level analyses concerning the step-wise integration of AI functionalities in ATs and railway lines that could possibly support the technological advancement into the rail sector have been discussed. Up to now, it seems that GoI2.1 would be the next step that could be achieved, especially by exploiting the concept of Safety Envelope. For GoI2.2 or higher, however, AI technical and certifiability issues would have to be resolved first.

Chapter 4

A Reference Methodology for Developing DL Applications

The view discussed in this chapter draws upon widely accepted practices in the DL community which could be deducted by accessing potentially any book on the topic (e.g., [52]); then, some of the discussed concepts may be familiar to researchers working closely with DL. The effort was mainly aimed at systematising the steps necessary for the development of DL applications, in order to design specific guidelines for the railway community.

4.1 Focus and Boundaries

The methodology is designed to identify an appropriate strategy when using sensors such as cameras and microphones. It emphasises the importance of understanding the specific characteristics of the target domain or problem, thus facilitating the definition of a *modular approach*. Each module would pursue a specific goal and could potentially be implemented through different techniques, both DL-based and non-DL-based; however, in the following, the focus is specifically on Deep Neural Networks (DNNs), which have proven to be quite efficient in the analysis of audio and video data (see Section 2.1.3). The assumption is that, when possible, a modular approach would always be preferable to introduce diversity, try to improve robustness, or facilitate the tasks that DNNs have to address.

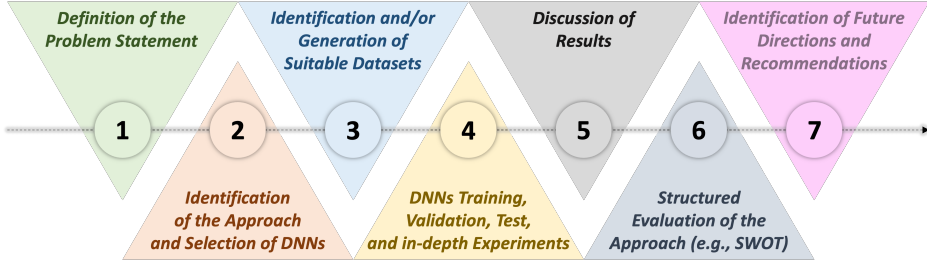


Figure 4.1. A 7-Step Methodology

As depicted in Figure 4.1, the proposed methodology consists of seven steps:

1. **Definition of the Problem Statement.** This would be the most important step: the relevant facets of the domain and/or of the problem to be addressed are identified, along with Research Questions (RQs) and Key Performance Indicators (KPIs), to clearly define the goal(s) to be achieved. A preliminary analysis of existing approaches is also performed.
2. **Identification of the Approach and Selection of DNNs.** A suitable, potentially modular, approach is established based on what was identified at the previous step. The selection of DNNs through which practically implement the approach should then follow.
3. **Identification and/or Generation of Suitable Datasets.** The datasets to properly train and test the selected DNNs are identified and/or generated, especially by taking into account the peculiarities identified at step 1.
4. **DNNs Training, Validation, Test, and in-depth Experiments.** DNNs are then trained, validated, and tested. In-depth experiments should also be carried out to identify the limits of the implemented models or threats to their validity.
5. **Discussion of Results.** Obtained results are analysed to evaluate the impact that the implemented approach could have. Also, possible

improvements should be investigated together with details about non-implemented functionalities.

6. **Structured Evaluation of the Approach.** To provide a clear picture of the implemented approach, together with its strengths and weaknesses, a structured evaluation is performed, where all relevant aspects are clearly highlighted and discussed.
7. **Identification of Future Directions and Recommendations.** Leveraging the lessons learned and the performed experiments, future directions and recommendations are identified, highlighting the aspects that could be of prominent importance for future developments.

The methodology is presented with examples drawn from the two case studies discussed in Chapters 5 and 6. However, the core principles of the methodology and the underlying concepts can potentially be applied to any other DL applications of this nature. The next section provides a detailed discussion of each step.

4.2 A 7-Step Methodology

4.2.1 Definition of the Problem Statement

The first step of the methodology expects to define and identify all the *facets* of the problem to be addressed, starting with a high-level definition of the objective(s) and arriving at the identification of the practical goal(s) to be pursued together with RQs and KPIs.

Objectives and Constraints. The *objective(s)* that is intended to be achieved should be defined by taking into account possible *constraints* delimiting the boundaries for the approach that could be adopted. The constraints include all the aspects that could preclude the utilisation of a specific approach/technique in favour of another.

Existing Approaches. An investigation of *existing approaches* oriented at addressing the same goal under the same constraints should be performed. If any, their methodology, strengths, and weaknesses should be evaluated

to understand how (and if) the proposed solutions can be exploited or improved.

Domain-specific and Problem-specific Characteristics. Together with the above, the *peculiarities* of the domain and/or of the problem (if any) that could help in developing a suitable approach should be identified:

- *Domain-specific Characteristics.* These represent peculiarities of the application domain (i.e., railway).

An example is given by the route of the trains against those travelled by road vehicles. The path of the train is constrained by rail tracks, while cars can *freely* move within their lane; the path of the train is predetermined (before the train starts its run) in most cases, while car drivers can dynamically change the path according to, e.g., traffic conditions; rail tracks are *always* delimited by rails, while lane markings may not be present or may be partially/totally erased. Hence, the *constrained route* is a *domain-specific characteristic* of the rail sector which can be exploited to set up suitable approaches to address problems in this field as better detailed in Chapter 5.

- *Problem-specific Characteristics.* These are related to the problem that is intended to be addressed.

For example, the problem analysed in Chapter 6 concerns the monitoring of Level Crossings (LCs) while leveraging only video and audio data. In this specific context, it can be assumed that the camera monitoring the LC is fixed (i.e., its field of view does not vary over time). The *fixed camera* is then seen as a *problem-specific characteristic* which has been exploited to facilitate the tasks that DNNs had to address.

Potential Challenges. The analyses above could highlight some **potential challenges** that, most likely, would be required to be addressed during the practical implementation of the approach. An example is given, as discussed in Chapters 5 and 6, by *data availability*.

Goals. All the evaluations above would contribute to the delineation of a clear picture of the problem to be addressed. Particularly, the analysis

of the existing approaches would suggest which are the aspects that could be addressed by simply re-using existing solutions and which, instead, would deserve further investigations. These considerations, together with the identified challenges, would set the basis for the identification of the *practical goals* that are intended to be achieved.

Research Questions and Key Performance Indicators. To conclude, it would be advisable to *formalise the problem statement* by means of *RQs* and *KPIs* which would respectively guide the research and identify criteria against which evaluate the implemented approach.

4.2.2 Identification of the Approach and Selection of DNNs

The second step expects to define a suitable approach to achieve the identified goals.

Identification of the Approach. The keyword in this context would be *modularity*. If the problem could be partitioned into different tasks, multiple modules can be identified each of which focuses on one of them. Sometimes, how to partition the problem could be deduced from the approaches implemented in the literature. Alternatively, the problem itself may have some characteristics that would make it subdividable; for example, as analysed in Chapter 6, LCs are composed of different warning/protection subsystems each of which can be monitored independently from others. Modularity would have four main advantages:

- *Adaptability.* In case a similar problem would have to be addressed, they can be substituted only some modules instead of having to engineer the whole approach from scratch.
 - *Tasks facilitation.* For example, object detectors can detect multiple objects at the same time, however, they may suffer from the small-scale problem (see Section 2.3). If the partition allows, expedients could be used to, e.g., adopt a classification approach instead of an object detection one.
 - *Redundancy.* Modules could “support” each other in making some decisions (as analysed in Chapter 5). This would be an extremely
-

important aspect as DL (but also AI in general) could suffer from some problems like *instability* (see Chapter 2) which could affect their robustness. Then, as also suggested within the RAILS survey [114], it would be advisable to design multiple modules implementing different solutions (both DL-based and non-DL-based) to address the same task so that, if one of them fails, there would be others as backup. If possible, also *sensor redundancy* could be exploited, by implementing a module for each sensor; this would further contribute to increasing the robustness of the final system.

To conclude, for each module, they should be defined the functionalities it should implement.

Selection of DNNs. Once the (possibly modular) approach has been identified, they should be selected the various techniques and models (including the DNNs that would be required to implement its functionalities.

With specific focus on DNNs, the a priori selection of the DNN that would perform the best on a given task is not that straightforward. The literature could be leveraged in this context, however, it would be advisable to first understand how the specific functionality could be implemented from a higher perspective, e.g., if through *classification*, through *semantic segmentation*, through *object detection*, and so on. Then, a specific DNN can be selected to start with by considering, e.g., the results it achieved in the literature; nonetheless, its viability should be always validated through experiments.

4.2.3 Identification and/or Generation of Suitable Datasets

The third step expects to identify and/or generate datasets to exploit for the implementation of the selected functionalities, specifically, to train and test the DNNs selected to implement them.

However, at this point, it is extremely important to underline a key aspect. There are cases where data have already been collected before even thinking that they could have been used, in the future, to build an AI system. In these cases, the *approach* (and related modules, functionalities, and DL models) would be most likely identified starting from the collected data, i.e., by understanding which would be the best strategy to extract

valuable knowledge from them; during this process, it would also be possible to consider the collection of additional data for the effective resolution of the identified problem. Having stated that, the following of this section discusses the opposite case, i.e., when data are collected ad-hoc to implement the functionalities identified in the previous step.

Dataset Identification. The first thing that could be done is to identify whether “suitable datasets” already exist. This is actually something that could be addressed by leveraging the analysis of existing approaches within the first step of this methodology.

Suitable datasets refer to datasets reflecting some specific characteristics (e.g., the domain- and problem-specific ones identified at step 1) that would be required for the correct implementation of the functionalities identified. In their absence, they should be generated as discussed below.

In any case, it is worth emphasising that, even though they are not completely suitable, existing datasets (if any) can still be leveraged. Examples are RailSem19 (adopted in Chapter 5) or AudioSet (adopted in Chapter 6), which have been respectively used to pre-train the DNN for rail tracks segmentation and to build a suitable dataset for warning bell detection.

Dataset Generation. This can be seen as a process including the following (optional or mandatory) activities:

- **Data Collection (*mandatory*).** As analysed in Section 2.4.2, data can be collected by: i) acquiring them directly on the field (*real data*); ii) synthetically generating them by means of 3D editors, real-life simulators, digital twins, and so on (*synthetic data*); or iii) extracting them from collections already available (which may contain both real and synthetic data). A combination of them is also possible. Important to mention, methods ii) and iii) above are more oriented at developing proofs-of-concept, i.e., understanding whether an approach could be potentially viable to solve a real problem without passing through the “on-field data collection”, which is typically the most time-consuming phase when it comes to develop AI models. Then, once the strategy has been tested and proved to be viable on *synthetic* data, its viability should be validated on real data [116].

- **Data Labelling (*mandatory/optional*)**. This would be *mandatory* in case of *supervised* approaches: data must be properly labelled according to the task and, consequently, to what the selected DL model would require in input. In Section 2.4.2, they have been presented some alternatives on how data labelling can be performed. On the other hand, data labelling could be *bypassed* if *unsupervised* approaches are adopted.
- **Data Pre-processing (*optional*)**. Another action that could be performed regards the application of specific transformations to data. In some specific cases, especially when adopting pre-trained networks, it would be advisable to pre-process data in the same way they were pre-processed to address the source task [25]. This would *not be mandatory*, however, pre-trained weights represent the knowledge acquired by processing data with some specific characteristics; respecting these characteristics would *probably* lead the DNN to achieve good performances with less effort.

In addition to that, there are other cases in which the transformations would completely change the type of data and/or where the samples collected should be subdivided into multiple samples to build the final dataset. This case is discussed in detail in Chapter 6: basically, data were pre-processed to pass from audios to spectrograms (which can be seen as images), and then, from each spectrogram, different patches were extracted to build the actual dataset.

- **Data Augmentation (*optional*)**. While the above transformations are oriented at modifying collected data, those expected by *Data Augmentation* would be oriented at increasing the number of collected data by (synthetically) generating additional samples. For example, data augmentation techniques could be used to add some specific events (e.g., atmospheric phenomena or specific light conditions) that were not captured when performing data collection.

Important to mention – besides *Data Collection* which is obviously the first action to be taken – *Data Labelling*, *Pre-processing*, and *Augmentation* are not intended to be exactly sequential. For example, *Data Pre-processing* could transform data that much to induce an adjustment of the previously

crafted labels; or, *Data Augmentation* could be performed before pre-processing data. These may change according to the selected approach.

4.2.4 DNNs Training, Validation, Test, and in-depth Experiments

Model Training, Validation, and Test are three established concepts that can be retrieved from about any book dealing with ML or DL (e.g., [9, 52, 152]).

Once datasets have been properly generated and/or identified, these are prepared by subdividing data (without repetitions) into Training, Validation, and Test sets. These will be used for the proper characterisation of the ML/DL model.

Training and Validation. With specific focus on DNNs, conceptually, the aim is to let the network learn from Training data the best set of weights that would help it to perform properly on data not contained in the Training set.

The training is performed in steps, named *epochs*. At each epoch, the DNN tries to learn from Training data and then, tries to perform predictions for the Validation data. The performances of the network are then evaluated at each epoch by taking into account these predictions. Epochs are typically not pre-specified in number, and the training could potentially never stop. To avoid that, some *stopping criteria* (e.g., the *Early Stopping*) can be adopted to stop the training when some conditions are met. Briefly, according to the *Early Stopping* criterion, the training will stop if the performances of the model evaluated on the Validation set do not improve for a pre-specified number of epochs (namely, *patience*). A trivial, widely adopted, strategy is the following: considering the loss function, which measures the prediction error that the model makes on a specific set, and assuming that the loss function has to be minimised, the training will stop when the loss function evaluated on the Validation set would not *decrease* for K (patience) epochs. This would also help to mitigate *overfitting*, i.e., it would prevent the DL model from sticking too much to the Training data so that it would not be able to correctly process data that are not contained in the Training set.

Worth underlining, as it has been a central aspect in the context of

this Thesis, the training can be performed according to the two main modalities already discussed in Section 2.1.2: *Training from scratch*, i.e., the weights of the network are randomly initialised and the network start extracting knowledge directly from Training data, and by applying *Transfer Learning*, i.e., some pre-trained weights learned on a different task are used to initialise network's weights.

To conclude with this high-level overview on DNN training and validation, during this phase, they are typically “tested” different network's configurations in terms of *parameters* (e.g., learning rate) and *hyperparameters* (e.g., number of layers/neurons). Typically, the training is repeated multiple times, each time by varying (i.e., *tuning*) parameters and hyperparameters, to try to find the configuration that would perform the best on Validation data. In this context, it would be possible (and advisable) to adopt strategies like the *k-fold cross-validation* [152] which could help to statistically evaluate the performance of the model during the training phase. Essentially, it expects to subdivide the dataset (excluding the test set) into k partitions: then, one of the partitions (or folds) will be used as the Validation set, the other $k-1$ folds will be used as the Training set. In this way, for each configuration, there will be k (commonly 10) different trainings, each of which is performed on slightly different Training data and evaluated on different Validation data. Conceptually, this serves to mitigate the bias that some specific data distributions could introduce: for instance, a too optimal Training set could lead to too high performance, conversely, another particular distribution could lead to low performance. By re-training the same configuration multiple (k) times, each time slightly varying the Training set (basically, exchanging one fold) and considering a different Validation set, it would be possible to retrieve the real potential of the network in relation to the considered dataset.

Testing. After having identified the *best configuration* (together with its *best weights*) the DNN should be tested on the Test set to evaluate its **generalisation performances**, i.e., to understand how it would perform on data it has *never seen* before. In the case of poor performance, it would probably mean that: i) the selected configuration is not adequate, then, others should be evaluated; or ii) the DNN, independently from the configuration, is not suitable to address the task, then, other DNNs should

be considered. In both cases, some of the actions discussed in previous steps should be re-performed to look for a suitable alternative.

In-depth Experiments. In case the model would perform properly on the Test set, additional experiments should be carried out to understand the limits of the model and identify the conditions (if any) that would affect the performance of the systems (e.g., evidencing possible *threats* to the actual *validity* of the model). These tests heavily depend on the task to be addressed and would be oriented at evaluating different aspects like, but not limited to, robustness to specific weather conditions (as evaluated in Chapter 5) or to environmental noise (as evaluated in Chapter 6).

4.2.5 Discussion of Results

The actions carried out in the above steps would lead to the implementation of the functionalities, whether based on DL or not, expected by the identified approach. Tests and related results should be analysed in detail to try to define (or forecast) the practical impact that the implemented functionalities would have in operational environments; particularly, shortcomings should be evaluated to understand if they would represent practical issues, and should then be addressed with high priority in the future, or some mechanisms exist that could be leveraged to mitigate them. In addition, in case not all expected functionalities have been implemented, they should be provided hints on how these could be developed and what their impact/contribution would be.

Eventually, the RQs and KPIs identified within the first step should be respectively answered and evaluated to formally underline the contributions that the conducted research has brought and the potential viability of the implemented approach.

4.2.6 Structured Evaluation of the Approach

Downstream the above analysis, it would be advisable to generate a sort of *identity card* of the implemented approach highlighting all possible aspects (both advantages and shortcomings) that should be considered for future improvements or possible comparisons with other approaches.

The SWOT Analysis

In the context of this Thesis, this is done by adopting one of the most known tools for the self-evaluation of Strengths, Weaknesses, Opportunities, and Threats (SWOT) of an implemented system from the market perspective, i.e., the **SWOT analysis** [6]. Notably, it is not used to evaluate market viability of the implemented approach, instead, it is used to structurally identify, indeed: i) the inner *strengths* and *weaknesses* of the approach in terms of technical aspects (e.g., accuracy and computation time); ii) the external *opportunities* that could support the implementation of such an approach compared to others; and iii) the external *threats* that, conversely, could affect its viability, performances, or practical applicability.

Considerations on the AI Act

Another possibility would be to evaluate the implemented approaches against some regulations or requirements tailored to the development of AI systems. Although these have not been the subject of the studies described in this thesis, they are extremely worth mentioning as it is now a matter of fact that the regulation of AI-aided systems is a central topic in the international panorama (as introduced in Section 2.4.1). An example in this direction is given by the emerging AI Act (simply referred to as Act in the following).

In Chapter 2 of the Act, some requirements are specified for *high-risk AI systems*, which conceptually include, among others, those that could have an impact on people's safety. Premised that ad-hoc regulations and standards would be required for the introduction of AI systems in railways, and considering the fact that if an AI system dealing with safety-critical functionalities cannot be rated SIL4 (or equivalent) it would not be allowed in operational environments, some considerations can be made, and some techniques could be adopted, to start moving towards some of the requirements posed by the Act.

For example, as for **data concerns**, in addition to what was already discussed in Section 2.4.1 (especially about *standardised detests*), another factor that should be considered regards the continuous monitoring of public areas, as would be the case if the Level Crossing monitoring system introduced in Chapter 6 were adopted in the field. In such scenarios,

sensitive information (e.g., people’s faces or cars’ license plates) might be recorded and, even though these specific data are not relevant for the AI model that is intended to be implemented, privacy concerns may arise as these data would be stored and passed to the model anyway. This would be a sensitive factor against which to evaluate the practical viability of the implemented approach. For the sake of knowledge, some techniques can be adopted to mitigate this issue: an example is given by *data anonymisation* approaches (e.g., [97]), which could be exploited to “obscure” sensitive data before storing them or using them for AI models training.

Another important aspect would be **transparency**. As mentioned in Section 2.4.1, the functioning of AI models, especially DL ones, may not be directly understandable by humans, which essentially means that the mapping between an input and the corresponding output may not be directly clear to the users. With respect to the case studies discussed in the following chapters, practical examples could be: i) understanding why the tree’s shadow is seen as an anomaly (see Section 5.4.2); or ii) evaluating why the LC barrier is not correctly detected when it is exactly vertical (see Section 6.4.2). An evaluation against *transparency issues* would then be required to be sure that the causes leading to any output of the AI would be comprehensible and evaluable. In this direction, *Explainable AI* approaches (e.g., SHAP [88], LIME [123]) could be investigated to understand, for example, which visual features contribute to the definition of a specific output, thus, potentially increasing the transparency of DL models.

These and other factors including (as per the Act) *technical documentation, record-keeping, accuracy, robustness, and cybersecurity*, would be of primary importance when developing AI systems to be adopted in operational environments. The case studies discussed in this Thesis have been analysed from a scientific perspective, with the aim of showing how they could be conceptually addressed. On the other hand, when aiming at developing systems to be deployed in real environments, in-depth analyses should be carried out to assess their market viability. To that aim, the requirements currently posed by the Act should be taken into account, nonetheless, the railway sector is governed by strict regulations and, with a focus on safety-critical applications (like the Obstacle Detection one discussed in Chapter 5), systems are required to be verified against rigorous safety standards to be deemed operational. Hence, although a

self-assessment against the Act may be possible, what would actually be required are ad-hoc AI railway standards defining how these requirements should be intended in railways and what practices should be adopted in turn to assess the SIL (or equivalent measure) of implemented AI systems.

4.2.7 Identification of Future Directions and Recommendations

Eventually, as the last step, based on the lessons learned and experiments performed, future directions and practical recommendations should be identified. These could serve as guidelines and suggestions for future research in the context of the developed approach and, possibly, also within a broader research field.

Case Study 1: Vision-based Obstacle Detection on Rail Tracks

As anticipated in Chapter 1, this case study was conceived as a *benchmark* to investigate the potential of some specific DL techniques, in combination with light onboard train equipment (i.e., cameras), for detecting any kind of obstacles on rail tracks (i.e., both known and unknown in advance). The analyses discussed in this chapter were then performed with the aim of demonstrating how the methodology defined in Chapter 4.2 could be applied to a practical Proof-of-Concept (PoC), trying to identify and discuss the main aspects that should be taken into account when dealing with such a problem and to define a suitable strategy for its resolution.

5.1 Problem Statement

Obstacle Detection (OD) would constitute one of the primary systems that the Intelligent Train Protection (ITP) (see Chapter 3) should implement to ensure safe run of Autonomous Trains in *open* environments. As mentioned in Section 2.3.1, OD systems may involve multiple sensors including LiDars, radars, and different types of cameras (e.g., RGB or thermal). Moreover, multiple Obstacle Detection systems installed as trackside equipment or on-board trains/drones (see Figure 2.4) can collaborate for

robust and long-range obstacle detection.

Premised that *sensor fusion* and the *cooperation between multiple OD systems* would be key factors for proper obstacle detection, this chapter focuses on analysing to what extent DL combined with data captured from cameras mounted on the head of a train could contribute to detect obstacles on rail tracks.

This focus is motivated by the central role that cameras have in object recognition (see Section 2.3.1) and by the cost-effectiveness of such sensors. Indeed, as also mentioned in Chapter 3, there are some railway lines where the Automatic Train Protection, or equivalent protection mechanisms, are absent. In these contexts, vision-based OD systems could play a key supportive role to affordably implement ITP and, thus, improve safety [44].

Objectives and Constraints. The objective would be to investigate how DL can be adopted to detect obstacles on rail tracks leveraging video data only.

Existing Approaches. Section 2.3.1 reports an examination of the literature on vision-based approaches in the context of OD on rail tracks. From those studies, it has been possible to identify a processing pipeline mainly involving three modules: *Rail Tracks Identification*, *Obstacle Detection*, and *Obstacle Distance Estimation*. One of the most evident findings is that OD has been mainly addressed with the implementation of *Supervised DL* approaches which, despite achieving high detection accuracy in most cases, may be limited in terms of *detectable obstacles* as only a set of pre-specified objects (on which the DL model has been trained) could be detected. Differently, *Unsupervised* approaches for the detection of (potentially) any obstacle on rail tracks have received limited attention. The same goes for obstacle distance estimation.

Domain-specific Characteristics. In the context of train driving, as partially anticipated in Section 4.2.1, the principal characteristic of the rail sector lies in the fact that the path of the train is constrained by rails and is typically pre-specified [154]. This peculiarity, also referred to as *constrained path*, has been exploited to set a particular strategy for rail track segmentation. Nonetheless, it led to the necessity of generating a

particular (suitable) dataset for that task. Additional details are given in Section 5.3.1.

Potential Challenges. The principal issue was a direct consequence of the *constrained path* characteristic just mentioned. Some datasets depicting real railway scene exist (e.g., RailSem19), however, none of them seem to reflect the characteristic above. Similarly, no dataset depicting obstacles on rail tracks was found. Then, *data availability* (widely discussed in Section 2.4.2) was the first challenge encountered.

Goals. The first goal concerned the generation of a suitable dataset based on the aforementioned characteristics. Then, the second goal was to evaluate whether such characteristics would have brought practical benefits in the realisation of a DL system for obstacle detection. Lastly, since plenty of supervised approaches have been proposed within the literature, but unsupervised ones oriented at detecting any possible anomaly on rail tracks received limited attention (Section 2.3.1), the third goal was to propose an unsupervised approach that, integrated with supervised ones, could have improved the coverage (in terms of localisable obstacles) of Vision-based On-train OD systems commonly investigated in the literature.

Research Questions (RQs) and Key Performance Indicators (KPIs).

These goals were then formalised through of the following RQs:

- **RQ1:** How would the mentioned domain-specific characteristic support or affect the implementation of a Vision-based On-train OD system?
 - **RQ2:** How could the problem of data availability be addressed? How can suitable datasets reflecting the characteristics mentioned above be generated (even synthetically) for the development of a PoC in this context?
 - **RQ3:** To what extent it would be possible to detect any kind of anomalies on rail tracks by leveraging unsupervised DL approaches and data coming from camera sensors only?
-

To conclude, **KPIs** have been identified for the high-level evaluation of the proposed approach:

- **KPI1: Obstacles Coverage.** Refers to the type and the number of obstacles that the system is capable of recognising and localising.
- **KPI2: Detection Distance.** Indicates the maximum distance at which the system is capable of detecting potential obstacles.
- **KPI3: Computation Time.** Concerns the time required by the system to detect possible obstacles.

5.2 Identification of the Approach and Selection of **DNNs**

5.2.1 A Modular Approach

The modular approach for the implementation of a Vision-based On-train **OD** system was drafted starting from the results obtained while analysing related literature (see Section 2.3.1). As mentioned, besides Rail Tracks Identification and Obstacle Distance Estimation approaches, most of the studies proposed Deep Neural Networks (**DNNs**) (mainly inspired by object detectors like YOLO) trained in supervised mode capable of detecting only some specific objects specified beforehand. In addition to that, for the sake of clarity and simplicity, it is worth recalling the following concepts already introduced in Section 2.3.1: “objects” refers to obstacles that are known and can be specified beforehand; “anomalies” refers to obstacles that are not or cannot be pre-specified, i.e., an obstacle whose type is unknown to the **OD** system; lastly, “obstacle” refers to both objects and anomalies.

In order to design an **OD** system potentially capable of locating all possible obstacles on rail tracks, the idea was to add two other components to the three mentioned and arrange them according to a modular architecture as that depicted in Figure 5.1. Hence:

- The *Rail Track Identification* module would, indeed, identify the tracks.
-

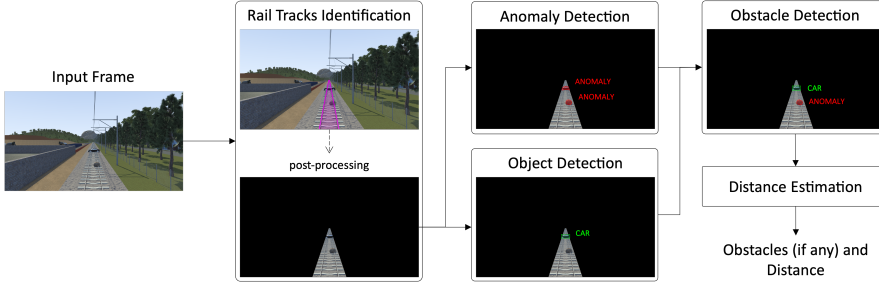


Figure 5.1. An Architecture for Vision-Based OD Systems

- The *Object Detection* module would be oriented at recognising and locating objects that could become potential obstacles for the running train.
- The *Anomaly Detection* module, in parallel to the above, would implement an Unsupervised approach to (potentially) locate any possible anomaly on rail tracks.
- The *Obstacle Detection* module would merge the outcomes from the precedent modules and set up the final prediction.
- Lastly, the *Obstacles Distance Estimation* would visionary estimate the distances of both objects and anomalies from the train.

Worth highlighting, this is one of the possible combinations that could be adopted. For example, it would be possible to put the first three modules in parallel, and then merge their results, e.g., by checking whether the detected obstacles overlap with the detected rail tracks. The important factor would be to have the OD modules supporting each other in the detection (or, at least, localisation) of any possible obstacle. Hence, the *Object Detection* module would focus only on the objects that are most likely to be found on/near rail tracks and would be extremely sensitive (like humans or vehicles), while the *Anomaly Detection* module would:

1. Support the prediction performed by the Object Detection module by, e.g., highlighting an anomaly in the same position as the object detected by the object detector.

2. Act as a *complementary* system by detecting anomalies that would not be detected by the Object Detection module.

5.2.2 Selection of DNNs

Rail Tracks Identification. As mentioned, this module focuses on identifying rail tracks. In Section 2.3.1, they have been identified different DL solutions that could be adopted for this purpose: *rail tracks detection*, *rail tacks segmentation*, and *rails identification*. Among them, those oriented at segmenting rail tracks were found to be (potentially) the most suitable to implement the proposed modular architecture for the following reason: they allow for the direct identification of the region of interest within which obstacles have to be identified (assuming that they are properly implemented). However, those studies propose effective as well as complex DL approaches for rail track segmentation and none of them seemed to mention the fact that the *constrained path* characteristic might have been exploited to address this task.

Then, to understand whether that characteristic would have actually brought benefits into rail track segmentation, it was adopted a simpler approach, compared to those identified in the literature, based on one of the most spread state-of-the-art DNNs when it comes to SS problems, i.e., U-Net [127].

To summarise, this module has been implemented for the following reasons: i) validate the fact that the *constrained path* characteristic could actually bring benefits in rail tracks identification; and ii) this module, according to the identified modular architecture, is propaedeutic to Obstacle Detection.

Object Detection. This module would be oriented at detecting a specific set of sensitive and most recurrent potential obstacles (like humans or vehicles). Given the large attention this task has received within the literature (Section 2.3.1), it has not been addressed in this Thesis. Nonetheless, it is important to mention that *object detection* is not a specific railway task: objects like pedestrians, vehicles, animals, and so on are actually common entities that could be identified in multiple scenarios. Then, to train or pre-train object detectors that could work also in railways, in case new ones would be intended to be proposed, it would be possible to leverage

datasets available online such as KITTI¹, MS COCO², and Pascal VOC³.

Anomaly Detection. This module is oriented at detecting anomalies on rail tracks. Inspired by some studies proposed in the literature (Section 2.3.1) [7, 146], the idea was to adopt an unsupervised DL approach based on image reconstruction. The implemented approach followed this rationale: i) a Deep Autoencoder is trained on non-anomalous frames only and learns how to reconstruct them; ii) since that, the DNN should not be capable of re-constructing a possible anomaly at run-time that, then, would be poorly reconstructed; iii) hence, a map locating the anomaly (i.e., anomaly map) can be produced by comparing the original image in input to the DNN with the reconstructed one; iv) lastly, the reconstruction error could also be taken into account to support the anomaly map (conceptually, the higher the error, the higher the possibility of having an anomaly). To be specific, to perform image reconstruction, a Vector Quantised-Variational Autoencoder (VQ-VAE) [146] was leveraged, whose loss function, inspired by [7], was slightly modified to introduce the Structural Similarity Index Measure (SSIM). Further details will be given in Section 5.4.2.

Obstacle Detection and Obstacle Distance Estimation. As mentioned, the Obstacle Detection module would simply merge the outputs from the aforementioned modules producing a single prediction for each obstacle in case both the Object and Anomaly Detection modules would highlight the same obstacle. This module would then produce a map that: i) identifies and classifies objects; and ii) locates anomalies.

The last module, i.e., Obstacle Distance Estimation, would then estimate the distance of the various obstacles from the train by applying both *class-based* and *class-independent* (see Section 2.3.1) approaches.

¹<https://www.cvlibs.net/datasets/kitti/index.php>

²<https://cocodataset.org/#home>

³<http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html>

5.3 Identification and/or Generation of Suitable Datasets

To train and test U-Net, two datasets were generated by taking into account the *constrained path* characteristic introduced below. Then, the [DNN](#) adopted to implement the Anomaly Detection module was trained and tested with datasets built by leveraging U-Net outputs.

5.3.1 Domain-Specific Characteristic: Constrained Paths

When training a [DNN](#), it is a good practice to evaluate its generalisation performances, i.e., understand how it would perform on data it has never seen during the training phase. This is mainly because it would not be always possible (or even nearly impossible) to build datasets depicting all the possible events that could happen in relation to a specific task. Then, the network is fed with the data that have been possible to collect; some of these are not included in the training/validation sets and are used to estimate, indeed, how the model would perform in case it would have to deal with unseen scenarios. In the case of vision-based approaches oriented at analysing, e.g., world scenes, this “issue” is even more sensitive as the environment is such aleatory that it would not be possible to represent in a *finite* dataset all possible events and facets that could occur.

In railways, and specifically for Rail Tracks Identification problems, the aleatory the [DNN](#) should face can be somehow reduced given the fact that the path is constrained by rails and is pre-specified. From a high-level perspective, two *types* of aleatory can be identified: *structural aleatory*, which is related to railway infrastructures (e.g., turnouts, level crossings, and railway stations along a rail track); and *environmental aleatory*, which is related to weather phenomena, illumination conditions, possible intrusions on rail tracks, and so on. While the latter is something that the [DNN](#) should address in any case, the former can actually be reduced.

Figure 5.2 shows two example lines: Line A is extremely simple, and is composed of a single rail track, while Line B has two rail tracks and, in the middle, there is a turnout that would allow a train to change track (introduces structural complexity). Despite a single “general purpose” [DNN](#) could be adopted to segment rail tracks in any condition, it would be

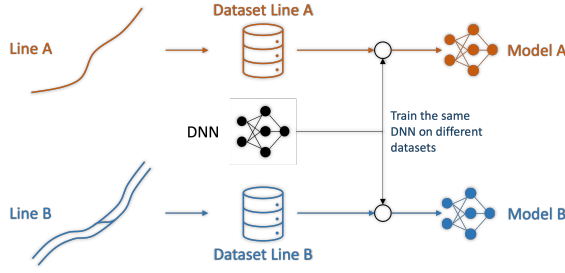


Figure 5.2. Building a DNN for each Railway Line

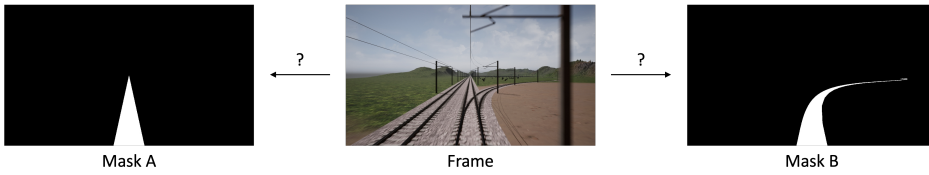


Figure 5.3. Which Mask should be Predicted?

possible to leverage the *constrained path* characteristic to build a dataset for each line so that the two resulting models (Model A trained on data from Line A, and Model B trained on data from Line B) would only have to deal with *environmental aleatory* and not with *structural aleatory* as railway infrastructures would not change so frequently. Then, by collecting multiple data, always from the same line, under different light and weather conditions, at run-time, the DNN would most likely have to process a scene which is extremely similar (yet not identical) to those it has already seen during the training.

In this context, there is also another aspect to consider. Figure 5.3 shows a particular scenario, which is actually common in railways: the running train is approaching a turnout. In this case, the extremely important question is: what is the rail track that has to be segmented? Assuming to have a DNN which has been trained with data coming from multiple lines and paths, and that is extremely efficient and reliable in segmenting rail tracks ahead the train, which is the “actual track ahead the train”? Would the model segment the straight railway track or would it segment the curved one? The concept above of building a DL model for each railway line can actually be refined further to: “build a dataset for each path the

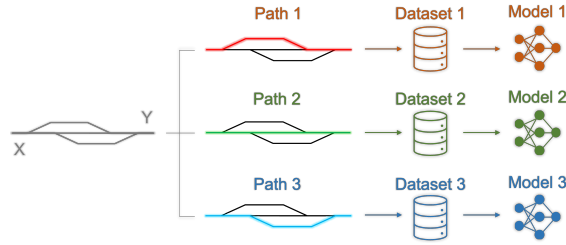


Figure 5.4. Building a DNN for each Path

train can travel”. A path, in the context of this Thesis, is the exact set of rail tracks that the train would travel to reach station Y from station X; Figure 5.4 graphically explains this concept. It has been demonstrated that, if the characteristic above is exploited and then the DNN is trained on data coming from a specific path only, it would be capable of predicting the correct mask (whether “A” or “B” in Figure 5.3) according to the pre-specified path it has to travel. Further details are given in Section 5.4.1.

Dataset Identification. There exist some publicly available datasets, like RailSem19 [165], that depict real railway scenarios captured from a camera mounted on the head of a train, nonetheless, they are collected from multiple tracks. RailSem19 has been exploited in the context of this study to pre-train U-Net and also implement a Self-Training approach for data labelling, however, suitable datasets had to be generated to validate the aspects mentioned above.

5.3.2 Virtual Railway Scenarios for Synthetic Data Generation

To build a dataset containing data collected from the same railway line (or path), RoadRunner and Unreal Engine were exploited to create virtual railway scenarios from which to extract those data.

Two virtual scenarios were developed, a simpler one with only one rail track in RoadRunner, and a more complex one with three rail tracks in Unreal Engine 5. Actually, RoadRunner was adopted in both cases to (at least) design the structure of the rail tracks.

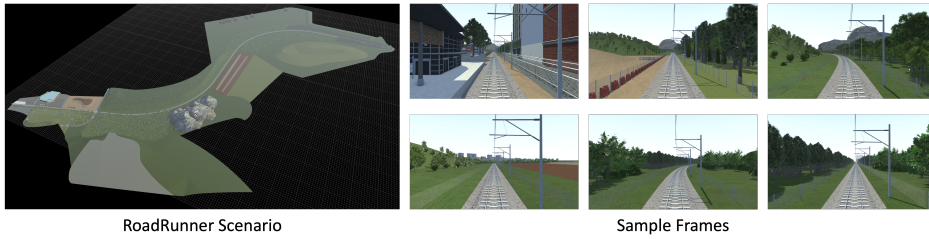


Figure 5.5. Scenario Generated in RoadRunner with Sample Frames

Virtual Scenario in RoadRunner

RoadRunner is a MathWorks tool conceived for the realisation of virtual scenarios for the experimentation of autonomous driving applications in automotive. Nonetheless, as shown in this context, it can be easily adapted to also generate railway scenes by either using some proprietary assets (intended as 3D models of the items to reproduce) or importing assets created through other 3D modelling software like Blender⁴.

Figure 5.5 provides an overview of the scenario generated in RoadRunner together with some example frames.

Virtual Scenario in Unreal Engine 5

Unreal Engine is a software conceived for the realisation of video games, nonetheless, it could be adapted also for scientific purposes. Its utilisation is not as trivial as for RoadRunner, however, it could allow for some other functionalities that facilitate the process of data generation, augmentation, and labelling. Indeed, it offers the possibility of simulating different weather and light conditions, as well as automatically generating labels (i.e., rail track masks).

The pipeline for the generation of synthetic data through Unreal Engine has been the following: i) Delineation of the rail tracks structure in RoadRunner; ii) Importation of the rail tracks into Unreal Engine; iii) Generation of the scene; iv) Video collection under different, simulated, weather and light conditions.

⁴<https://www.blender.org>

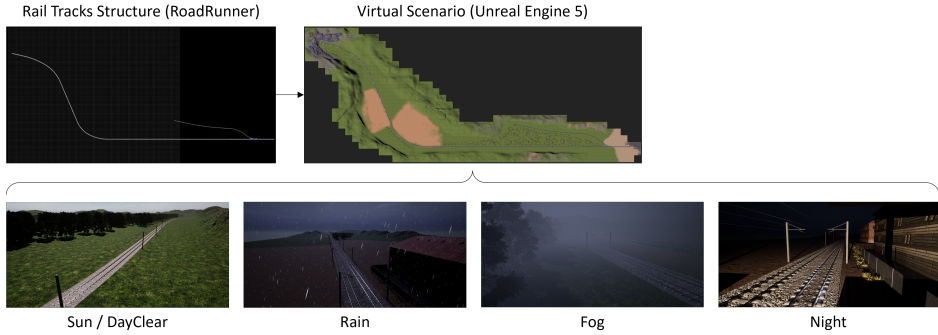


Figure 5.6. Overview of the Scenario Generated in Unreal Engine 5

Figure 5.6 provides an overview of this pipeline and the scenario generated in Unreal Engine. As mentioned, this railway scene is a bit more complex than the former, with two parallel rail tracks and a “side” rail track branching off the “main” one.

5.3.3 Generation of Suitable Datasets

Once the virtual scenario was created, three datasets were extracted to train and test U-Net. One was extracted from RoadRunner, and required also the implementation of a Semi-Automatic Labelling algorithm to accelerate the labelling process; conversely, the other two were extracted from Unreal Engine and did not require any labelling effort as it was automatically performed in Unreal Engine.

RoadRunner Dataset: RR Dataset

Data Generation. Two videos were extracted from RoadRunner: a video that depicts the scene free from any anomaly (“FreeTrack” video) and a video depicting a scene where a car is obstructing rail tracks (“WithCar” video). The first one was leveraged to build the dataset that has been used to train and test U-Net, the second one, instead, was adopted to evaluate how the models would hypothetically have behaved at run-time.

Then, 2000 frames were extracted from the FreeTrack video and processed as follows to generate the RR Dataset.

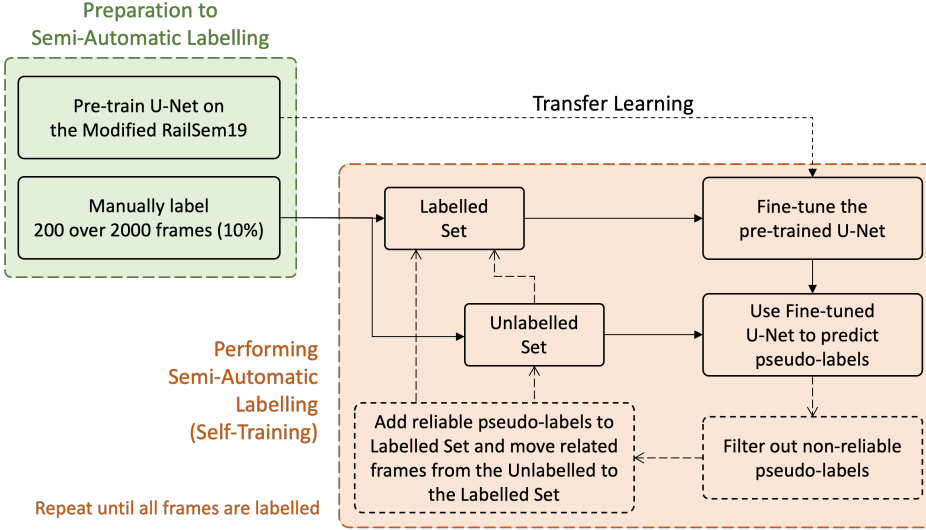


Figure 5.7. Semi-Automatic Labelling based on Self-Training and TL

Data Labelling. Not to hand-label all 2000 frames, and thus to accelerate the labelling process, a Semi-Automatic Algorithm was implemented leveraging the concepts of Self-Training and Transfer Learning (TL).

Before introducing it, however, a brief digression on RailSem19 is required as it was crucial for the application of TL. It is composed of about 8500 images properly labelled for semantic segmentation, nonetheless, it encompasses labels for multiple objects like roads, poles, tram tracks, and so on. Hence, some Python scripts⁵ were exploited to keep only the masks related to the rail tracks and discard all images not containing them. Additional details on this process are given in reference [115]. Eventually, it was obtained a “Modified RailSem19” dataset composed of 5519 images (and related masks) that were split into Training (3863 images; $\sim 70\%$), Validation (828 images, $\sim 15\%$), and Test (828 images; $\sim 15\%$) sets.

Figure 5.7 shows an overview of the implemented Semi-Automatic Labelling algorithm whose steps are detailed herein:

⁵https://github.com/xmba15/rail_marking/tree/master/scripts

1. *Preparation to Semi-Automatic Labelling:*

- (a) Pre-train U-Net on the modified version of RailSem19. For the sake of consistency, additional details on the pre-training will be given in Section 5.4.1 when U-Net training, validation, and test are illustrated.
- (b) Partially label the 2000 frames extracted from RoadRunner. In particular, considering sequential frames, one frame each 10 was manually labelled in order to have an even distribution of labelled frames over the entire rail track. Hence, only 200 (10%) were manually labelled (through LabelMe⁶).
Then, initially, the Labelled Set and the Unlabelled Set were composed of 200 and 1800 images respectively.

2. *Performing Semi-Automatic Labelling.* At each step:

- (a) Augment the Labelled Set by horizontally flipping images and related masks.
- (b) Split the Augmented Labelled Set into Training and Validation.
- (c) Fine-tune the Pre-trained U-Net on the Training set until the loss function evaluated on the Validation set does not improve for more than 20 epochs (Early Stopping).
- (d) For each image in the Unlabelled Set, as shown in Figure 5.8:
 - i. Use the Fine-tuned U-Net to predict the pseudo-label (the mask) for the image.
 - ii. Use the Fine-tuned U-Net to predict the mask for the flipped image.
 - iii. Average (pixel by pixel) the predictions.
 - iv. Check if the generated pseudo-label is “reliable” (mechanism explained below). If so, add the image and the label to the Labelled Set.
- (e) Repeat from 2.(a) until all frames are labelled.

⁶<https://github.com/wkentaro/labelme>

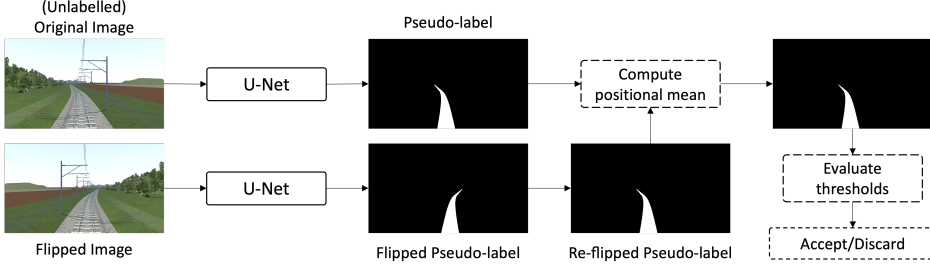


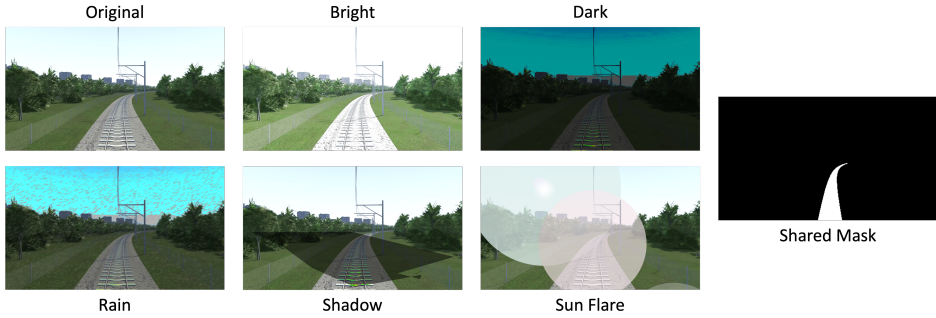
Figure 5.8. Evaluation of Generated Pseudo-labels

In-depth details can be found in [115], nonetheless, it is worth clarifying how the generated pseudo-label would have been deemed “reliable”.

In general terms, Self-Training (e.g., [167]) expects to generate a “teacher” which, learning from labelled data, produces some *pseudo-labels* for the unlabelled data; then, a “student” model is trained by combining data with both hand-crafted labels and pseudo labels. For trivial tasks, a *pseudo-label* can be considered “potentially correct” if, e.g., the confidence the teacher associates with it is higher than a certain threshold. For example, in an image classification task between cats and dogs images, considering a threshold of 95%, if the teacher predicts that the image X contains a cat with confidence equal to 99%, then, the image is associated with the label “cat”. Nonetheless, in this case, where U-Net makes a prediction for each pixel producing a score between 0 (background pixel) and 1 (rail track pixel), this would not have worked properly. Then, the strategy in Figure 5.8 was adopted. Once the two predictions have been merged by positional averaging the values that U-Net predicts for each pixel, a “new” pseudo-label was obtained. Then, for each pixel in the new pseudo-label, if the predicted value was lower than 0.05 (bottom threshold), then, it was considered as actual background; exactly the opposite was done with rail tracks pixels, considering a threshold value of 0.95. If the averaged value was in between these two thresholds, the pixel would have been classified as “uncertain”. To conclude, if the number of uncertain pixels was less than 0.1% compared to the total number of pixels in the image, the pseudo-label would have been considered “reliable” and added to the Labelled Set. This strategy could also have been applied directly to a single prediction, i.e., without considering the flipping. Nonetheless, the flipping added a second

Table 5.1. Data Augmentation Transformations (Transf.)

Transf.	Description
Bright	Randomly increase the brightness of the image
Dark	Randomly decrease the brightness of the pixel
Rain	Simulate rain by randomly adding drops with different intensities and angulation
Snow	Simulate snow by randomly increasing the brightness of specific pixels
Shadow	Simulate shadows by adding random semitransparent masks on the image
SunFlare	Simulate sun flares by adding concentric and random circles in the image

**Figure 5.9.** Examples of Transformations on Frame from RoadRunner FreeTrack Video

prediction to each pixel making the whole process more robust to possible completely wrong predictions.

Downstream this process, all 2000 frames were labelled. Nonetheless, to meet GPU memory constraints (after having done some tests), only 1600 of these 2000 were selected which were then augmented as follows.

Data Augmentation. Data Augmentation was applied by leveraging the Automold Python library⁷. This offers algorithms for different kinds of transformations including those summarised in Table 5.1 which were exploited in this case. Notably, all transformations introduced some randomness, e.g., “Shadow” introduces transparent polygons with random shapes in random positions.

Then, for each of the 1600 *original* frames, all transformations in Table 5.1 were applied; Figure 5.9 shows their effects. Worth underlining,

⁷<https://github.com/UjjwalSaxena/Automold--Road-Augmentation-Library>

Table 5.2. Distribution of Samples in *RR Dataset*

Set	<i>Original</i>	<i>Bright</i>	<i>Dark</i>	<i>Rain</i>	<i>Shadow</i>	<i>SunFlare</i>	Total
<i>Training</i>	1280	1280	1280	1280	1280	1280	7680 (80%)
<i>Validation</i>	160	160	160	160	160	160	960 (10%)
<i>Test</i>	160	160	160	160	160	160	960 (10%)
<i>Total</i>	1600	1600	1600	1600	1600	1600	9600

the mask generated for the original frame (“shared mask” in the figure) was used also for the transformed frames derived from it.

Then, the final *RR Dataset* was composed of 9600 frames which were then subdivided into Training, Validation, and Test sets as shown in Table 5.2.

To conclude, it is worth underlining that this dataset has been published on Zenodo⁸.

Unreal Engine Datasets: MainTrack and SideTrack Datasets

Generating datasets in Unreal Engine is easier from the perspective of data labelling, as it can be done automatically by the software itself, but is quite harder in terms of scenario generation: Unreal Engine offers wider customisation compared to RoadRunner, but is not as user-friendly as RoadRunner.

A total of 8 videos were collected through Unreal Engine whose frames i) *depicted rail tracks without any anomaly* and ii) were automatically labelled during the extraction. Four of these videos were collected while the train “was running” on the *Main Rail Track*, while the remaining four were collected while the train “was running” on the *Side Rail Track*. The videos are categorisable as: i) *DayClear*, depicting the scene in the case of a sunny day; ii) *Rain*; iii) *Fog*; and iv) *Night*, which also involved the simulation of a train headlight illuminating the rail tracks. All these environmental conditions were simulated directly in Unreal Engine. Figure 5.10 explains the difference between the two rail tracks and shows some frames related to them.

From each video, 1500 frames were selected, for a total of 6000 frames

⁸<https://zenodo.org/records/7924875>

5.4 DNNs Training, Validation, Test, and in-depth Experiments

This section discussed the training, validation, and test phases for the DNNs adopted to implement the Rail Track Identification and Anomaly Detection modules. Worth underlining, U-Net, adopted for rail track segmentation, has been evaluated on all the datasets presented in the previous section. Conversely, the VQ-VAE adopted to perform the anomaly detection has been trained with data from the RR Dataset once these were pre-processed through U-Net.

5.4.1 Rail Tracks Identification: U-Net

As mentioned in Section 5.2.1, a SS approach leveraging U-Net (adapting a PyTorch implementation available on GitHub⁹) was implemented to identify rail tracks.

Pre-training on Modified RailSem19

As the first step, which was exploited also to implement the Semi-Automatic Labelling, U-Net was pre-trained on the modified version of RailSem19 obtained as indicated in Section 5.3.3.

U-Net was trained by considering an Adam Optimizer with a learning rate equal to 10^{-4} , a loss function based on the Binary Cross-Entropy loss, and the Dice Score as the evaluation metric. The Dice Score, however, is generally computed as the fraction between the double of the Intersection over Union (IoU), computed comparing the actual mask with the predicted one, and the total number of pixels contained in the two masks. This, however, could have resulted to be biased towards the “background class” as, in a mask of this kind, the majority of the pixels were related to the background. Then, to properly evaluate the performances of the network, the Dice Score was calibrated on white pixels only, i.e., those related to the “rail tack”, which is the subject to be identified. Then, the Dice Score was computed as:

⁹https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/image_segmentation/semantic_segmentation_unet

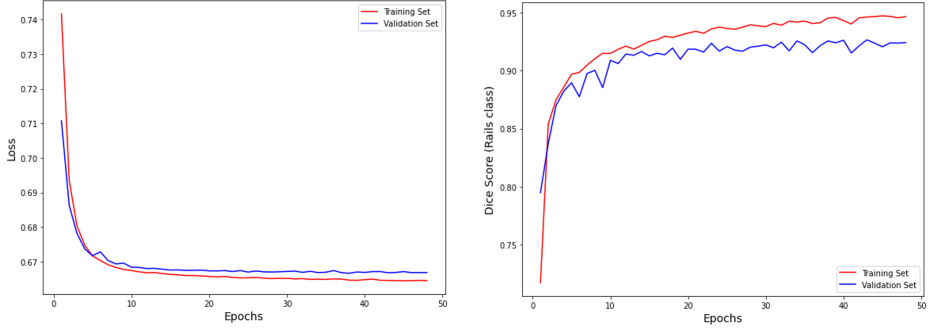


Figure 5.11. U-Net Pre-training on Modified RailSem19: Training Curves

$$Dice\ Score = \frac{2 \times \text{Overlapping White Pixels}}{\text{Total Number of White Pixels in both Masks}}$$

Having stated that, the training was performed *from-scratch* by setting a maximum number of 100 and adopting an Early Stopping criterion with patience equal to 10: if the loss evaluated on the Validation Set did not decrease for 10 consecutive epochs, the training would have stopped. Also, images were re-sized from 1920x1080 to 480x270 and a batch size of 8 was considered for the feasibility of the training. Figure 5.11 shows the training curves, i.e., loss and Dice Score trends, evaluated on the Training and Validation sets.

The network reached the lowest validation loss at the 38th epoch, achieving the following performances: training loss (i.e., value of the loss function evaluated on the Training set) and validation loss equal to 0.6646 and 0.6666 respectively, with a training and validation Dice Score of 0.9455 and 0.9257 respectively.

To conclude, U-Net was tested on the Modified RailSem19 Test Set to evaluate the generalisation performances on this dataset achieving a Dice Score of 0.9183.

The network showed promising performances even without a meticulous tuning of parameters and hyperparameters. Then, the set of weights related to the 38th training epoch on the Modified RailSem19 Test Set were stored and taken into account in the following tests to apply Transfer Learning.

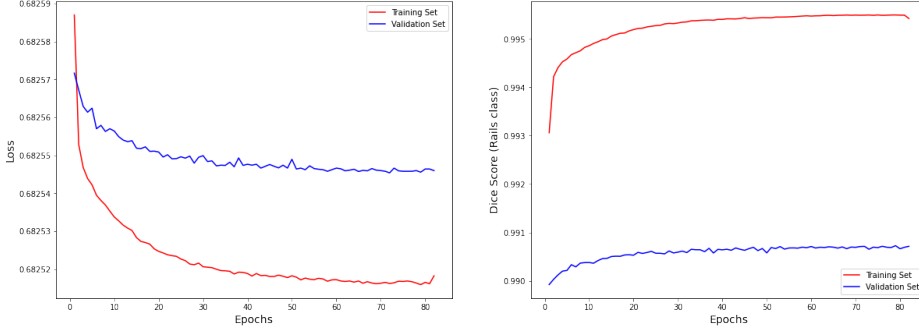


Figure 5.12. U-Net Training Curves on RR Dataset

These weights will be referred to as “pre-trained weights” in the following of this section.

Training, Validation, and Testing on RR Dataset

Exactly the same settings (Early Stopping, learning rate, etc.) mentioned above were considered to fine-tune the pre-trained U-Net on the RR Dataset.

Training and Validation. The best performances were achieved at the 72th epoch with: a validation loss and a training loss respectively equal to 0.682516 and 0.682545, while the validation and training Dice Scores were equal to 0.9907 and 0.9954.

Typically, they would be required more in-depth experiments by varying parameters and hyperparameters to achieve a (sub-)optimal solution. Nonetheless, the network surprisingly achieved good performances with the “base” settings introduced above. It also resulted to be quite stable, i.e., training curves in Figure 5.12 do not present heavy oscillations; worth underlining in this direction, after the first few epochs, the validation Dice Score did not fall under 0.99 for the whole training.

Generalisation Performances on Test Set. Once trained, U-Net achieved a Dice Score of 0.9993 on the RR Dataset Test Set. It was also checked whether a specific transformation of those indicated in Table 5.1

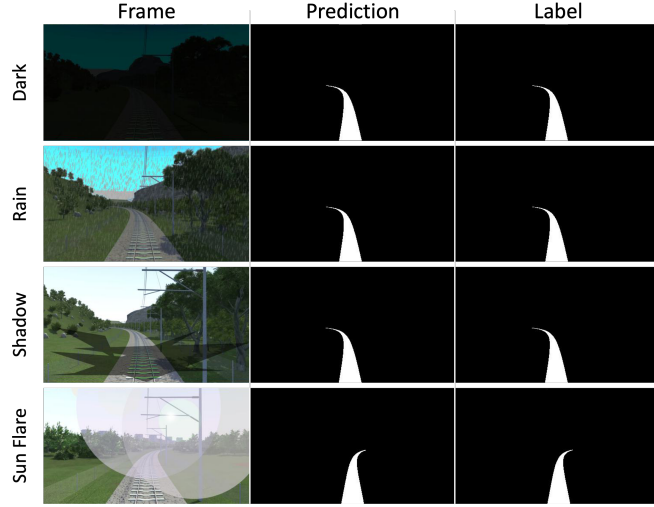


Figure 5.13. U-Net Predictions on RR Dataset Test Set Frames

would have particularly affected U-Net performances by separately evaluating the predictions: only in the case of “Rain” the Dice Score was equal to 0.9989, in all other cases, it was higher than 0.9992. Figure 5.13 shows the predictions for some augmented frames in the Test Set.

These results show that U-Net is quite suitable to address such a task under the conditions set out by the generated scenario and the considered augmentations. Additional details can be found in [115].

Training, Validation, and Testing on Unreal Engine Datasets

As explained above, two datasets were generated from Unreal Engine. These evaluations were performed especially to validate the hypothesis set out in Section 5.3.1:

*By collecting a dataset for each railway path and training the same **DNN** on these datasets separately, different **DL** models could be obtained, each of which is “specialised” in detecting rail tracks from a specific path. Then, before the train starts its run, it would be possible to “load” the **DL** model trained with data coming from the path it is going to travel. Hence, in case of “problematic frames” like that shown in Figure 5.3, the **DL** model would be capable of detecting the correct “rail track ahead the train”.*

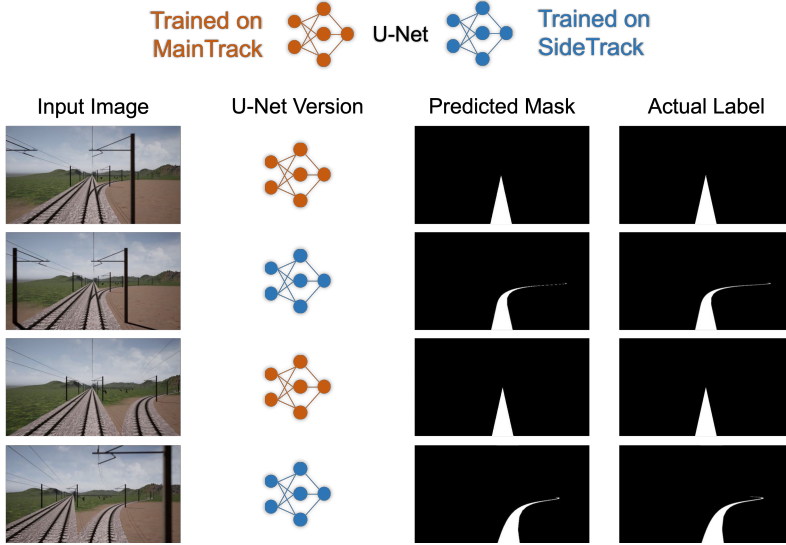


Figure 5.14. U-Net Correct Prediction of Rail Tracks Ahead the Train: MainTrack and SideTrack Datasets

Figure 5.14 graphically shows the idea above which has actually been proven to be possible. Indeed, if U-Net was trained on the MainTrack dataset, it detected the straight rail track; conversely, if it was trained on the SideTrack dataset, it detected the curved rail track.

Speaking of numerical results, U-Net was trained on both the MainTrack and SideTrack considering about the same settings as in the case of the training on RR Dataset (also loading the pre-trained weights):

- *Training, validation, and testing on MainTrack.* U-Net achieved the lowest loss on the Validation Set at the 30th training epoch. The training and validation losses at that epoch were 0.681584 and 0.681585 respectively; while the training and validation Dice Scores were equal to 0.9672 and 0.9668 respectively. Then, on the Test Set, the network achieved a Dice Score of 0.9668. “Transformation-by-transformation” tests were performed also in this case, clearly, taking into account data of the Test Set; the Dice Score was greater than 0.9664 in any weather and light condition.

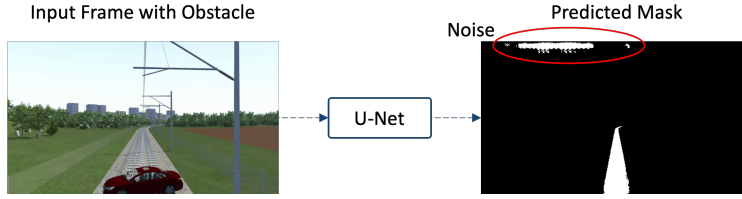


Figure 5.15. U-Net Noisy Prediction of Frames with Obstacle

- *Training, validation, and testing on SideTrack.* The best loss on the Validation Set was achieved at the 67th epoch. The training and validation losses were 0.681515 and 0.681521 respectively, while the Dice Scores were respectively equal to 0.9658 and 0.9659. The Dice Score evaluated on the Test Set was equal to 0.9659. As for the tests on different transformations, the Dice Score was always greater than 0.9658.

Despite these tests may not be enough to make general claims, and despite the fact that the performances of U-Net on these datasets might be slightly improved to match those achieved on the RR Dataset, it can be stated that, in these conditions, the network is capable of detecting the right rail track ahead the train if the domain-specific characteristic is exploited.

To conclude, the dataset generated through Unreal Engine served to validate the hypothesis above. All other tests and activities described from now on have been performed by taking into account videos and frames extracted from RoadRunner.

Testing U-Net on Anomaly Data

Since U-Net was trained with non-anomalous data only (i.e., with images that did not depict any obstacle on the rail track), some tests were performed to understand how it would have worked in case of obstacles. To that aim, it was exploited the *WithCar* video introduced in Section 5.3.3. The main problem, as depicted in Figure 5.15, was that the obstacle introduced some noise in some of the predicted masks. Important to mention, the noisy phenomenon could have been exploited to state that the frame was anomalous and then that, most likely, an obstacle was lying on rail tracks.

Nonetheless, this solution is not that robust as noisy events could happen also for other causes, e.g., adversarial attacks [98]. Hence, a post-processing algorithm was implemented to clean noisy predictions and retrieve the correct mask for rail tracks.

U-Net Post-processing

To filter out the noise, as explained in detail in reference [115], it was proposed the following *cleaning algorithm*. This, at the same time, filters out the noise and “enlarges” the predicted mask to include in the segmentation also the sides of the rail track.

The algorithm expected to:

1. *Filter out the macro noise.* Since the DL model would be adopted to extract rail tracks always from the same path, which does not change that often in terms of infrastructure, it would be possible to set up a *cleaning mask* that is the result of the bit-wise AND between all the possible masks that could be obtained from all the frames extracted from that path. Hence, at run-time, it could be possible to perform the bit-wise AND between the predicted mask and the *cleaning mask* to filter out the macro noise.
2. *Filter out the micro noise.* Some residual, micro noise could survive the above screening. To remove it, some functions of the *OpenCV* library¹⁰ could be exploited to extract polygons within the predicted mask and keep only the largest one, i.e., that related to the rail track.
3. *Enlarge the mask.* The trained U-Net highlights as rail track pixels those that are in between the rails and those belonging to the rails. This choice was made to remain in line with the labels that were extracted from RailSem19. Nonetheless, even if an obstacle does not lie exactly on the rail tracks but is close to them, it could be hit by the shape of the train; then, it could represent a threat and has to be detected. To enlarge the mask to include the sides of the rail tracks, it was exploited the polygon including the rail tracks extracted at the previous step. Essentially, the point where rails vanish at the

¹⁰https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

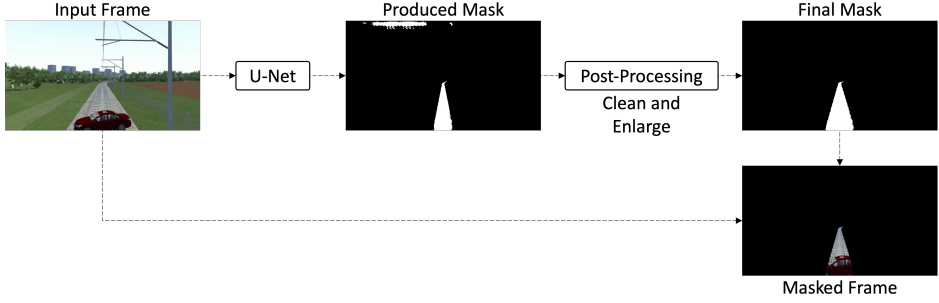


Figure 5.16. U-Net Pipeline: Prediction and Post-Processing

horizon was extracted; this was then used as the “pivot” to perform some rotations and produce rotated masks; these were hence added to the cleaned mask to obtain the enlarged one.

All steps above are explained with more details in reference [115]. The data to train and test the [VQ-VAE](#) were then generated by processing the RR Dataset through the pipeline shown in Figure 5.16.

Before proceeding with that, however, there are some aspects to highlight that could help to improve the post-processing algorithm for future implementations:

- First, the *cleaning mask* was obtained by merging all the masks predicted by U-Net on the *WithCar* video frames, which were actually noisy. Then, a mask was empirically drawn and used as the cleaning one. To improve this and obtain a tight *cleaning mask*, it would be possible to merge all the masks (i.e., the labels) contained in the dataset (RR Dataset in this case) which are “noiseless by construction”. This is because, in the end, both the RR Dataset and the *WithCar* video were obtained from the same train path; hence, the tight cleaning mask obtained in this way would have perfectly worked also to clean the masks predicted for *WithCar* frames.
- Second, train U-Net to directly predict larger masks such as to avoid the enlargement in post-processing. This, however, would be a little bit more challenging for the following reasons. Rails are almost always visible, hence, the network would theoretically be capable of easily

detecting them; that is not true for ballasts or sleepers that, in the case of, e.g., snow, could be not visible. Then, tests should clearly be done to understand how the network would behave in case it has to directly predict a “larger” mask; in case it does not perform properly, the solution presented above could be exploited.

5.4.2 Anomaly Detection: SSIM-VQ-VAE

The Anomaly Detection module was implemented by exploiting an *image reconstruction* approach based on VQ-VAE [146] and by considering the following rationale: train the VQ-VAE on non-anomalous data so that, at run-time, if an anomaly occurs it would be poorly reconstructed and then detected.

Datasets to Train, Validate, and Test VQ-VAE

As mentioned in the above section, the datasets to train, validate, and test the implemented VQ-VAE were built starting from the RR Dataset. Specifically, all the frames (4143) extracted from the “Free Track” (see Section 5.3.3) were considered. These frames were:

- A. Directly processed through the pipeline in Figure 5.16.
- B. First augmented by applying the transformations in Table 5.1, and then processed through the pipeline in Figure 5.16.

All the frames were then cropped to obtain 256x256 frames with the rail track aligned with the bottom centre to obtain squared images matching the input of the implemented VQ-VAE; an example of these frames can be seen in Figure 5.17. Then, two datasets were obtained: point A above led to the construction of the *NonAnomaly* Dataset, while point B led to the construction of the *AugmentedNoAnomaly* Dataset. Notably, to build the latter, only 1500 (over 4143) frames were considered; which were then augmented as already mentioned. This was necessary not to have a too large dataset to meet some GPU memory constraints: practically, as a general rule, the larger the dataset the longer the training time, especially if the batch size cannot be increased otherwise the maximum capacity of the GPU would be exceeded. This would be explained better below.

Table 5.4. Distribution of Samples in *NonAnomaly* and *AugmentedNoAnomaly* Datasets

Set	NoAnomaly	AugmentedNoAnomaly						Total
		Original	Bright	Dark	Rain	Shadow	SunFlare	
<i>Training</i>	3315 (80%)	1200	1200	1200	1200	1200	1200	7200 (80%)
<i>Validation</i>	414 (10%)	150	150	150	150	150	150	900 (10%)
<i>Test</i>	414 (10%)	150	150	150	150	150	150	900 (10%)
<i>Total</i>	4143	1500	1500	1500	1500	1500	1500	9000

Table 5.4 reports the subdivision of the samples in the two datasets above into Training, Validation, and Test sets. Worth mentioning, also these datasets have been published on Zenodo¹¹ together with the RR Dataset.

Explaining the SSIM-VQ-VAE

The implemented SSIM-VQ-VAE was inspired by [7] and [146].

Some tests were performed by implementing the solution proposed by [7]¹², nonetheless, given that the images in the dataset were mostly black, with a small portion occupied by rail tracks, it was like that the network reached a sort of plateau by predicting full-black images; indeed, the loss was quite low and remained constant over time. Then, experiments were performed by taking into account a Vector Quantised-Variational Autoencoder [144] as proposed in [146] exploiting an implementation found on GitHub¹³ while taking advantages of the benefits that the Structural Similarity Index Measure could bring in unsupervised anomaly detection if included as part of the loss function [7].

Both the architectures mentioned above have an encoder-decoder shape, however, in the case of VQ-VAE, the output of the decoder is first processed by an embedding algorithm whose *Embedding Space* is automatically learnt during the training. As exemplified in Figure 5.17, features in output from the decoder are arranged as 3D matrices. The algorithm substitutes the depth vectors (“ z_i ” in the figure) with the nearest centroid in the Embedding

¹¹<https://zenodo.org/records/7924875>

¹²<https://github.com/plutoyuxie/AutoEncoder-SSIM-for-unsupervised-anomaly-detection->

¹³<https://github.com/nadavbh12/VQ-VAE>

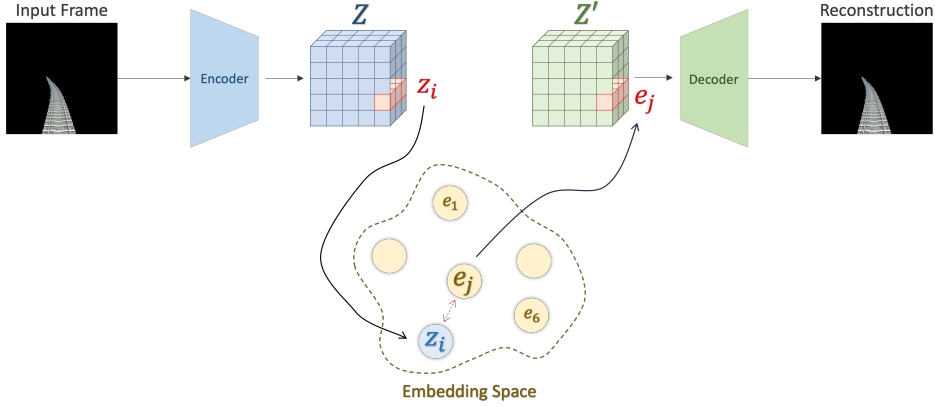


Figure 5.17. Insights on VQ-VAE [144]

Space (“ e_j ” in the example), then, the processed features are presented in input to the encoder. The number of centroids in the Embedding Space is finite and passed as a parameter of the network; also, centroids are updated (i.e., learned) during the training. The sort of *discretisation* introduced by the Embedding Space seemed to favour the training leading to well-reconstructed images.

Another important aspect to consider concerns the modification of the original VQ-VAE’s loss function [144]. Without going deeper with the details, the loss functions include different contributions related to the error made in reconstructing the image in input and some aspects related to the learning of the Embedding Space. The latest were left unchanged, while the former was substituted to pass from a binary cross-entropy loss evaluating the pixel-by-pixel error, to SSIM-based one inspired by the results obtained in [7]. To be specific, the new component was computed as 1 minus the mean SSIM score obtained by comparing the image in input with the reconstructed image. This score is a number between 0 and 1, where 1 means that the two images are identical.

Extremely important to mention, which is also another key factor that led to the choice of the SSIM-based loss function, is that the just mentioned score is computed by averaging all the values obtained by comparing the two images, region by region through convolutions, setting a squared filter to weight the computations (which then become a parameter of the network).

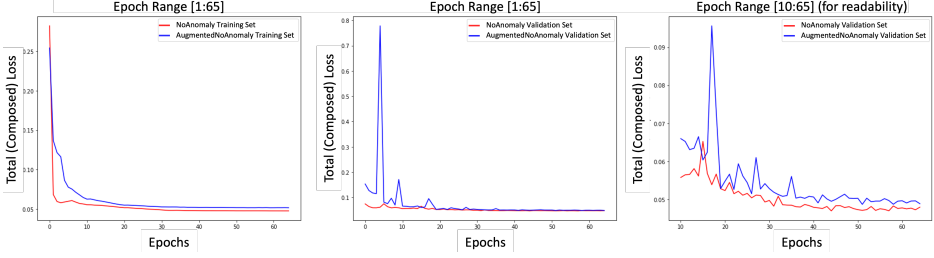


Figure 5.18. Training **SSIM-VQ-VAE** on *NoAnomaly* and *AugmentedNoAnomaly*: Losses

This convolution, stopping a step right before the mean operation, leads to an “image” that, from a high-level perspective, highlights the differences between the input image and the reconstructed one; therefore, the *anomaly map* is computed directly by the **SSIM-VQ-VAE** bypassing the need of making this comparison in post-processing.

Additional, in-depth details on these concepts are given in [115].

Training, Validation, and Testing on AugmentedNoAnomaly and NoAnomaly Datasets

SSIM-VQ-VAE was trained by setting an Embedding Space with 512 centroids and considering an 11x11 circular-symmetric Gaussian filter [148] to weight the **SSIM** computations. Then, it was adopted an Adam Optimizer with a learning rate equal to 10^{-4} , a maximum number of epochs of 100, and an Early Stopping criterion with a patience of 20 that monitored the composed loss function. The **SSIM** score was instead used as an evaluation metric as it provides information about the reconstruction capabilities of the **DNN**. Lastly, the batch size was fixed to 6 as, besides the network’s weights, the GPU had also to store the Embedded Space; a higher batch size would have resulted in a memory overflow.

The network was trained on both the *NoAnomaly* and *AugmentedNoAnomaly* datasets to study the contribution brought by the augmentations. Figures 5.18 and 5.19 report the training curves of **SSIM-VQ-VAE** in terms of loss and **SSIM** score respectively. In the case of training on *NoAnomaly* Dataset, the lowest validation loss was achieved at the 45th epoch; while, in the case of training on *AugmentedNoAnomaly* Dataset,

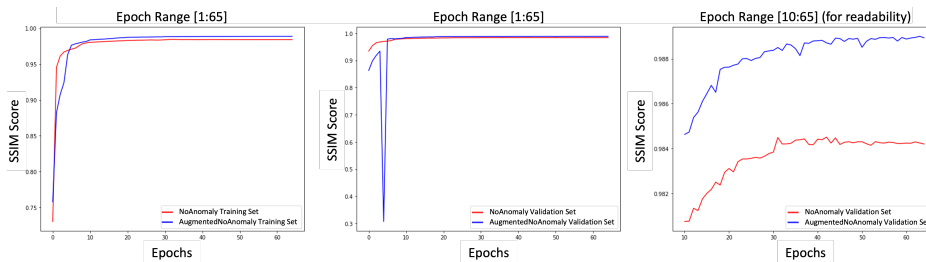


Figure 5.19. Training **SSIM-VQ-VAE** on *NoAnomaly* and *AugmentedNoAnomaly*: **SSIM** Scores

the lowest validation loss was achieved at the 52nd epoch. Then, the performances of the **DNN**, trained on the two datasets, were evaluated on the corresponding Test Sets. Table 5.5 reports the numerical results.

Figure 5.20 graphically shows some reconstruction results. As it can be noticed, in the case of *NoAnomaly* Dataset, the reconstructions were characterised by a sort of red patina (this happens with all frames). Then, it was performed a cross-test to understand how the **DNN** trained on the *NoAnomaly* Training Set would have performed on augmented data by evaluating its performance on the *AugmentedNoAnomaly* Test Set; this also served to understand which of the two datasets led to the model that generalised the knowledge the best.

Hence, Figure 5.21 compares the results obtained on the *AugmentedNoAnomaly* Test Set, transformation by transformation, by the **DNN** when trained on the *NoAnomaly* Training Set and when trained on the *AugmentedNoAnomaly* Training set. As expected, performances were higher in the second case even on the original samples (i.e., those without augmentation).

Table 5.5. Performances of **SSIM-VQ-VAE** at the Best Epoch(s)

	Performances on NoAnomaly Dataset	Performances on AugmentedNoAnomaly Dataset
<i>Best Training Epoch</i>	45	52
<i>Loss on Validation Set</i>	0.047049	0.048762
<i>SSIM Score on Validation Set</i>	0.984478	0.988777
<i>Loss on Test Set</i>	0.046928	0.049173
<i>SSIM Score on Test Set</i>	0.984483	0.988714

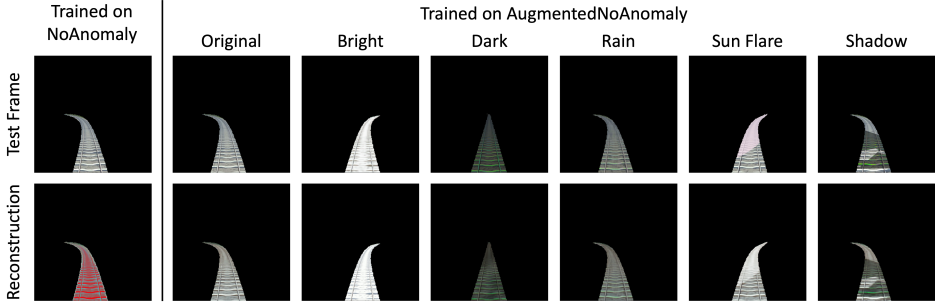


Figure 5.20. SSIM-VQ-VAE Reconstructions on *NoAnomaly* and *AugmentedNoAnomaly* Test Data

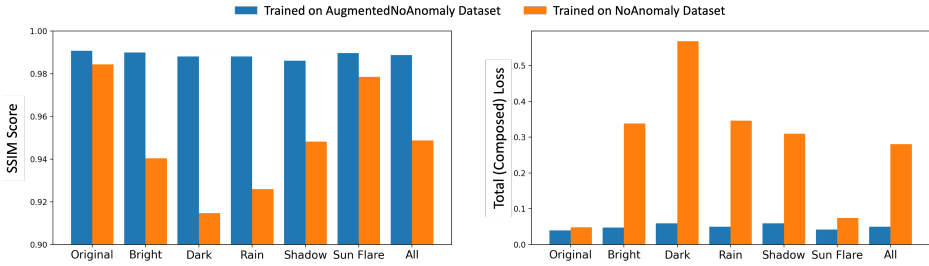


Figure 5.21. Performances of SSIM-VQ-VAE trained on *NoAnomaly* (orange) and on *AugmentedNoAnomaly* (blue) evaluated on the *AugmentedNoAnomaly* Test Set

Despite these tests may not be comprehensive enough to generalise the findings, it can be stated that, under these conditions, the augmentations allow the DNN to perform better in all the tested cases, therefore, the following analyses were performed by considering the SSIM-VQ-VAE trained on the AugmentedNoAnomaly.

Testing on Frames with Obstacles

As the last test, it was studied if the trained SSIM-VQ-VAE would have been exploitable to locate obstacles on rail tracks. To that aim, the “WithCar” video frames were adopted, after that they were pre-processed through the pipeline in Figure 5.16. Since the pre-processing through the pipeline basically masks the original frames in input to highlight only the

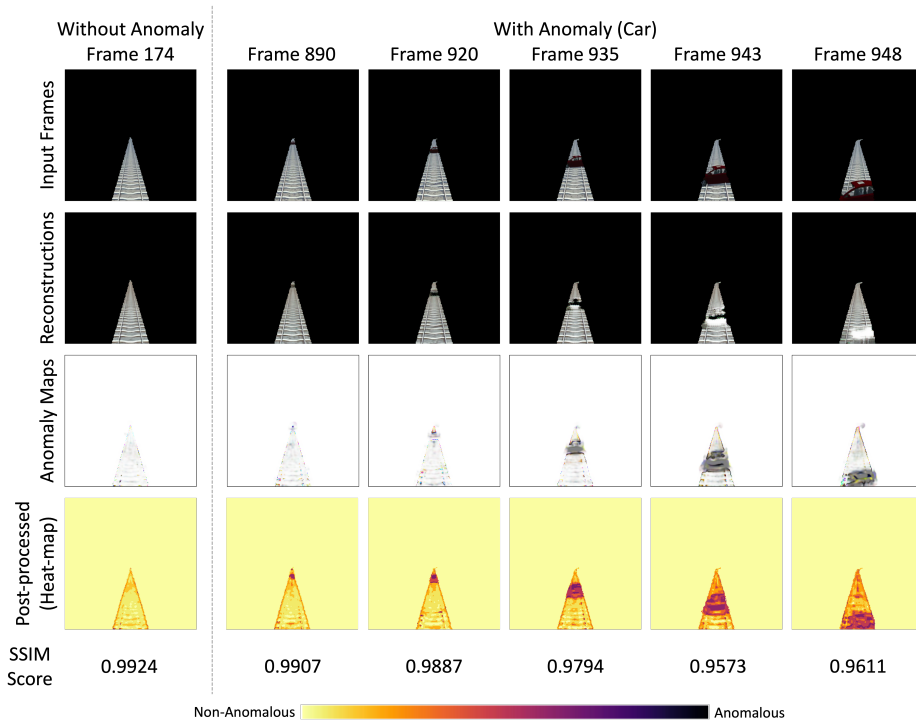


Figure 5.22. Examples of Anomaly Maps Generated through [SSIM-VQ-VAE](#) and related Post Processing (Heatmaps)

rail tracks, the pre-processed frames that are then fed to the [SSIM-VQ-VAE](#) are referred to as “masked frames” in the following.

Figure 5.22 shows some of the anomaly maps generated by the [DNN](#). Notably, a post-processing algorithm based on the K-Means clustering algorithm¹⁴ was applied to cluster together similar pixels to reduce the noise in the predicted anomaly map; then, it was generated a heatmap that better highlighted the anomaly. As can be noticed, the area occupied by the car was highlighted as an anomaly, however, there was still some noise disturbing the prediction.

Table 5.5 shows that the [DNN](#) has a quite low loss value on the Aug-

¹⁴<https://www.kdnuggets.com/2019/08/introduction-image-segmentation-k-means-clustering.html>

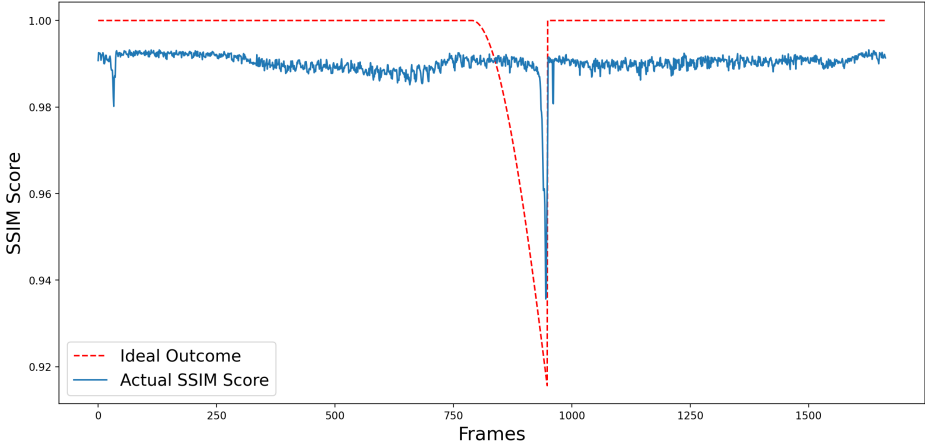


Figure 5.23. SSIM Scores Generated by SSIM-VQ-VAE on *WithCar* Frames

mentedNoAnomaly Test Set and, at the same time, the SSIM score indicates that the reconstructed image is extremely similar to the image in input (on average). This would actually mean that, in the case of non-anomalous frames, the anomaly map and, consequently, the heatmap (last line of images in Figure 5.22) should have been completely (or almost) empty. However, with the post-processing adopted, the heatmaps obtained were rather noisy, whose noise seemed to highlight little anomalies where there were none.

In addition to that, there are other three aspects to consider, two of which can be retrieved from Figure 5.23 which reports the SSIM Score evaluated on all the 1663 frames (in sequence) of the *WithCar* video. Ideally, the SSIM Score should have been like that indicated by the dashed red line (drawn just as an example), i.e.: equal to 1 until the car starts to appear within the masked frame produced by U-Net (around frame 790); hence, it should have started decreasing until the moment right before the “collision” (around frame 948) where it should have reached the lowest value. The actual SSIM Score broadly follows this trend and it suddenly drops just a few frames before the “collision”. This can be due to the fact that the anomaly map is too wide compared to the anomaly when the car is “distant” from the train; also, the majority of the masked frame is black, and that part is perfectly reconstructed rising, consequently, the value of the SSIM

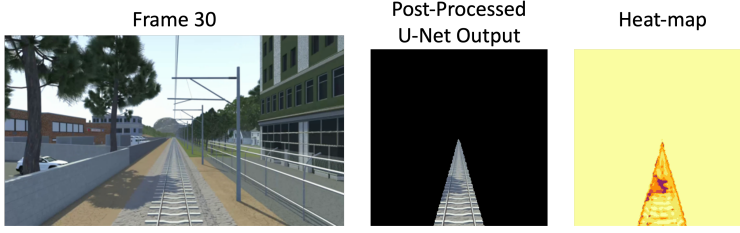


Figure 5.24. SSIM-VQ-VAE False Positive given Tree's Shadow

Score. This basically means that the SSIM Score starts dropping only when the car is so close to the train that it occupies a significant portion of the masked frame; nonetheless, it is quite correctly highlighted in the heatmap. This basically proves that, in these conditions, the SSIM Score is not that reliable to identify whether a frame is anomalous or not.

As the second aspect retrievable from Figure 5.23, it can be noticed that around frame 30 there is another sudden drop of the SSIM Score, even though it is way smaller than that related to the car. The motivation is shown in Figure 5.24, i.e., the shadow cast by one of the trees is highlighted as an anomaly. As said, the drop in the SSIM Score is not significant, but the anomaly in the heatmap is quite evident. In this case, the SSIM Score would have helped to filter out the False positive while the heatmap would have led to the activation of protective measures.

The two cases above, then, led to an evident problem: in some cases, the SSIM Score could be leveraged, in others not; in some cases, the heatmap correctly highlights the anomaly when the SSIM Score does not, in other cases it highlights something that is not actually a threat.

All the aspects are of crucial importance for the effective detection of anomalies and would surely require further and in-depth investigation.

5.5 Discussion of Results

The section above discussed the implementation of the U-Net and SSIM-VQ-VAE that were respectively adopted to segment the rail tracks and highlight anomalies. Figure 5.25 graphically summarises the implemented pipeline; some complementary analysis could be found in reference [115].

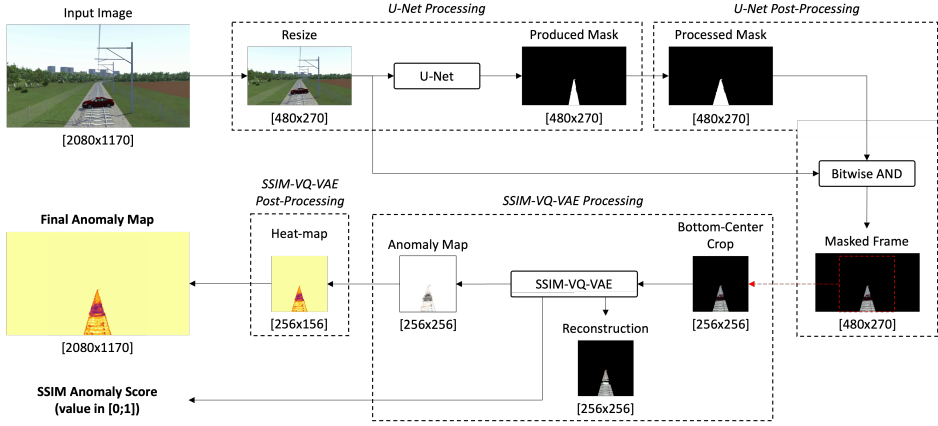


Figure 5.25. Rail Track Segmentation and Anomaly Detection Pipeline

5.5.1 Analysing Results and Future Implementations

Considering the performed tests and the augmentations presented above, here are discussed the main criticalities and opportunities for future improvements related to the implemented pipeline.

Rail Track Identification Module

This module encompasses both U-Net and the related post-processing. U-Net was trained and tested with data extracted from RoadRunner and Unreal Engine, achieving good results in segmenting the rail tracks within video frames.

The experiments performed on the data generated through Unreal Engine were oriented at understanding if, in complex scenes with multiple rail tracks, U-Net would have been capable of highlighting the correct rail track ahead the train. The results demonstrated that, if U-Net is trained with data coming from a specific path, it would be able to achieve what is stated above.

Then, all other tests were performed by considering data coming from RoadRunner. In particular, the tests carried out on the *WithCar* video highlighted that, in case of anomalies, the mask produced by U-Net would have been quite noisy and some post-processing was required to adjust the

prediction.

In addition to these tests, however, there are some other considerations to make, always taking into account the *WithCar* video whose frames were processed sequentially from frame 1 (“departure” of the train) to frame 1662 (the latest captured frame). Important to underline, the car (i.e., the obstacle) starts appearing within the video around frame 631; about 424 meters, empirically measured in the virtual scenario, before the “collision”. A human driver would potentially have been capable of seeing the car starting from that point, even though the car is quite imperceptible. Nonetheless, the mask produced by U-Net for that frame and some subsequent ones do not contain the car. The car starts appearing within the masked frames around frame 790 (about 180 meters to collision).

Actions that could be taken to improve these results include:

- Perform a more meticulous labelling process, trying to improve, if needed, the Semi-Automatic Labelling algorithm or manually labelling all frames.
- Try not to downscale the images that much. At the current stage, images are down-scaled to 480x270 pixels before being put in input to U-Net. Together with the image, also the label is down-scaled and the segmentation may get absorbed by the background, especially at the horizon when rail tracks are quite thin. This, however, would require more powerful hardware.

To conclude, U-Net takes, on average, ~ 20.6 ms to process a single frame; the post-processing takes ~ 363 ms, which is quite high. The post-processing, then, should be revised to reduce computation times. Some suggestions in this direction have been already given in Section 5.4.1. Additional details can be retrieved by consulting reference [115].

Anomaly Detection Module

To implement anomaly detection, an approach based on image reconstruction was investigated. To that aim, the [SSIM-VQ-VAE DNN](#) was implemented, where the [SSIM](#)-based loss function allowed the [DNN](#) to directly produce the anomaly map. Then, a post-processing based on K-Means clustering was adopted to reduce the noise of the anomaly map

and produce a heatmap that would have better highlighted the anomaly. Nonetheless, this introduced a series of issues, discussed in the previous section, leading to the conclusion that a new kind of post-processing should be implemented to try to improve the results. On the other hand, the [SSIM-VQ-VAE](#) achieved quite high performances in reconstructing non-anomalous images.

In addition to the above, also in this case, there are some considerations to make in relation to the detection distance and the computation time.

First, as mentioned above, the detection distance also depends from the segmentation produced by U-Net. However, masked frames start containing the car from about frame 790, while the anomaly highlighted by the anomaly map starts to become “clearly” visible around frame 890, i.e., about 70 meters to the collision. For those masked frames, however, the [SSIM](#) Score is still quite high (~ 0.99); this mismatching should be somehow further investigated.

Second, [SSIM-VQ-VAE](#) takes, on average, ~ 38.9 ms, while the post-processing through the K-Means clustering takes ~ 53.8 ms. This further suggests the investigation of a different post-processing algorithm. Unfortunately, at the current stage, post-processing does not seem to be bypassable, as the anomaly maps are rather noisy, not excessively, but just enough to require a minimum of intervention.

In order to try to overcome these issues, together with the False positives due to the shadows as mentioned in the previous section, the following preliminary directions were identified:

- Increase the dataset by adding more “problematic” frames (e.g., including shadows or sun flares) to let the [DNN](#) learn how to properly handle these events.
 - Enlarge the latent space. This is the core of the [VQ-VAE](#) and, theoretically, a larger latent space could allow more detailed reconstructions.
 - Use larger input sizes or crop images differently. 256x256 images were used to train the [DNN](#) where most of the pixels were black. Possibly, tests with higher dimensions, maybe also cropping the masked frames so that the rail tracks would occupy more space in the cropped one, could be carried out.
-

As for the Rail Tracks Identification module, further details on these aspects can be found in reference [115].

5.5.2 Answering RQs and Evaluating KPIs

Given the above experimentation, this section tries to formally answer to the RQs and evaluate the KPIs set in Section 5.1.

Research Questions

RQ1: How would the mentioned domain-specific characteristic support or affect the implementation of a Vision-based On-train OD system?

Experiments showed that the *constrained path* characteristics, presented in Section 5.3.1, could actually be exploited to teach DNNs to detect the correct rail track ahead the train. This might be a bit more onerous to implement than a model oriented at detecting all rail tracks in the scene, as: i) a dataset should be collected from each path the train could travel; ii) the selected DNN should be trained multiple times, one for each path, originating multiple models that should be stored somehow; iii) each time the train is starting a run, the right model should be loaded. Nonetheless, in the conditions set by the generated scenarios, it has been shown that the presented strategy would be viable. In addition to that, there would also be the advantage of reducing the *structural* aleatory the model would have to address at run-time (see Section 5.3.1).

RQ2: How could the problem of data availability be addressed? How can suitable datasets reflecting the characteristics mentioned above be generated (even synthetically) for the development of a PoC in this context?

Premised that real data would always be preferable, as the systems should deal with reality at run-time, not everyone has the possibility of accessing data collected on the field. In addition to that, putting real obstacles on rail tracks would not be that safe, unless experiments are conducted on a test track, which is even more complicated to obtain than real data. There exist some datasets available to the research community (e.g., RailSem19 [165], OSDaR23 [134]), nonetheless,

they seem not to reflect the domain-specific characteristics identified. Hence, 3D Editors, specifically RoadRunner and Unreal Engine, have been shown to be valid allies in generating synthetic datasets to train and test DNNs; those allowed to set up a strategy that resulted to be viable, especially in the context of rail tracks segmentation. However, the real applicability should always be validated on data from operational environments.

RQ3: To what extent it would be possible to detect any kind of anomalies on rail tracks by leveraging unsupervised DL approaches and data coming from camera sensors only?

The implemented strategy seems to be promising but would require further adjustments. Approaches based on image reconstruction, which have also been investigated in the literature (see Section 2.3.1), seem to be valid candidates for detecting any anomaly on rail tracks. In the context of this Thesis, tests were performed by considering a car as the anomaly, and the model was empirically found to be able to detect it about 70 meters before the potential collision.

Key Performance Indicators

KPI1: Obstacles Coverage. *Refers to the type and the number of obstacles that the system is capable of recognising and localising.*

The application of unsupervised DL approaches in a vision-based OD system could help to improve its coverage in terms of detectable obstacles. Nonetheless, further tests should be performed to effectively specify the type of detectable anomalies in terms of, e.g., dimensions or position.

KPI2: Detection Distance. *Indicates the maximum distance at which the system is capable of detecting potential obstacles.*

The performed tests showed that the evaluated anomaly starts to be visible in the heatmap at about 70 meters from the potential collision. Assuming that further investigations will be performed and that the approach would be perfected to have a “reliable” model that would be capable of precisely detecting the anomaly within those 70 meters, the applicability of the approach in a real environment would always

depend on the specific case that has to be faced. The stopping distance is conceptually given by the combination of two main elements: i) drivers' reaction and ii) the actual braking distance of the train which depends on factors like speed, mass, wheel/track adhesion conditions, and so on [39]. Both these elements are variable, and especially the latter depends on the target scenario; hence, evaluations should be performed case-by-case. For instance, high-speed trains or heavy freight trains typically require hundreds of meters to stop completely, nonetheless, there could be other scenarios where detecting an obstacle 70 meters in advance could be sufficient to ensure safety while applying automation. An example is given, as anticipated in Chapter 3, by railway lines which are currently not protected and where trains proceed at reduced speed. Alternatively, this system can be adopted to implement a sort of "return to home" functionality: in case of malfunction, the train could enter the "return to home" state and move autonomously towards the next station proceeding at reduced speed. The specific requirements against which evaluate the effectiveness of the approach in terms of detection distance should then be analysed in detail once the target operational scenario is defined.

KPI3: Computation Time. *Concerns the time required by the system to detect possible obstacles.*

In total, the implemented pipeline in Figure 5.25, takes about 476 ms, which is not exactly in line with real-time constraints. Nonetheless, the greater contributions are given by the post-processing algorithms, as evaluated in Section 5.5; optimising them would therefore be one of the primary steps to be taken to reduce computation time.

5.6 Structured Evaluation of the Approach

This section summarises the relevant aspects discussed in this chapter in relation to the adopted approach. These, together with other interesting facets, are arranged by following the structure of the **SWOT** analysis (see Figure 5.26) so that they can be easily evaluated.

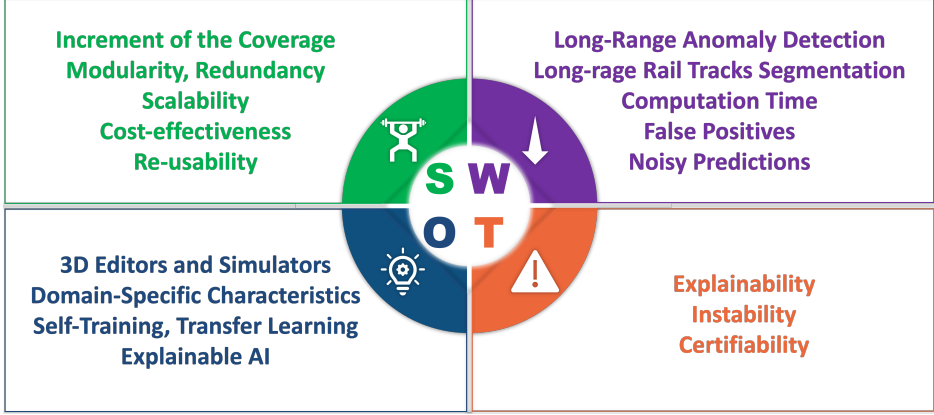


Figure 5.26. A SWOT Analysis for the *Rail Track segmentation and Anomaly Detection Pipeline*

5.6.1 Strengths ($\uparrow$) and Weaknesses ($\downarrow$)

Increment of the Coverage ($\uparrow$). The vision-based obstacle detection task has mainly been investigated with the application of supervised DL (see Section 2.3.1). The proposed approach, instead, expects to combine both supervised and unsupervised approaches to, indeed, improve the coverage of detectable obstacles (either known/specifiable in advance or not) on rail tracks.

Modularity, Redundancy, and Scalability ($\uparrow$). The *modularity* of the proposed approach allows for the implementation of this strategy also to already developed (supervised) OD applications, without compromising their performances. Indeed, the pipeline in Figure 5.25 could be juxtaposed to any supervised OD model without interfering with it in any manner.

Modularity also introduces a sort of *redundancy*, as anticipated in Section 5.1, since, for each known obstacle, two predictions would be generated: one by the Object Detection module and the other by the Anomaly Detection one. These modules would be implemented through different DNNs, making it reasonable to think that at least one of them would be capable of locating the possible obstacle.

Redundancy could be further improved given the *scalability* of the approach, as it can be easily replicated for each used camera in case of, e.g., stereo vision. Also, it could work with different types of cameras (e.g., thermal or RGB). This would basically ensure multiple predictions for a single event, reducing the probability of miss-detection.

Cost-effectiveness and Re-usability (↑). If properly implemented, the proposed solutions could actually bring to a cost-effective increment of safety in some specific scenarios as only cameras, and no other expensive sensor, would be required. Those scenarios may include, as mentioned above and in Chapter 3, autonomously reaching the nearest station in case of train malfunctions that prevent it from operating at maximum efficiency. Alternatively, a system developed through the proposed approach could act as a low-speed complement in main lines.

In addition to that, the approach, developed once, could also be re-implemented in other scenarios as the idea at the base would hold in any case obstacles have to be detected through camera sensors only. An example is given by level crossings, where the approach can be exploited to check whether the area is free from any obstacle.

To conclude, a camera mounted on the head of a train can also be exploited for other purposes, e.g., detecting defects in rail tracks or catenary components (see Section 2.3.2). Hence, cost-effectiveness may also refer to the fact that, once installed, the sensors can be exploited for multiple purposes.

Long-Range Anomaly Detection and Computation Time (↓).

These aspects have already been widely discussed in the previous sections. At the current stage, the implemented pipeline (in Figure 5.25) takes about 476 ms to process a single frame and anomalies can be detected within a range of about 70 meters. Although there could be applications where these performances might, probably, be enough (if combined with high accuracy), further investigations are required to optimise results.

Long-range Rail Tracks Segmentation (↓). As discussed in Section 5.5, the implemented U-Net correctly segments rail tracks in a medium-long range, nonetheless, it struggles in segmenting the part of the rail tracks that is far from the train, which typically is quite thin. The potential causes leading to this phenomenon have been discussed in Section 5.5 together with some actions that could be taken to mitigate them. This aspect is crucial to investigate, as if rail tracks at the horizon are not segmented correctly, it would not be possible to detect anomalies well in advance.

False Positives and Noisy Predictions (↓). To conclude, tests showed that the SSIM-VQ-VAE is prone to False positives as an anomaly a shadow cast by a tree standing in the proximity of rail tracks was identified as an anomaly. In addition, the post-processing applied to better highlight anomalies actually emphasises also the noise. These aspects would require to be deeply investigated and addressed in future research, especially studying the combination between the anomaly highlighted in the heatmap and the SSIM Score produced by the DNN as mentioned in Section 5.4.2.

5.6.2 Opportunities (↑) and Threats (↓)

Domain-specific Characteristics (↑). Some domain-specific characteristics could help to implement a suitable strategy and, at the same time, facilitate the task that DL models have to address. In this chapter, it has been taken into account the *constrained path* characteristic that, if exploited, could lead to the implementation of suitable semantic segmentation models capable of identifying the correct rail track ahead the train even in scenes where multiple rail tracks are present. Additional details on this have been given in Sections 5.3.1 and 5.4.1, and within the answer to RQ1 (Section 5.5.2).

3D Editors and Scenario Simulators (↑). The main problem that was identified in Section 5.1, even before starting implementing the DNNs, was the *lack of suitable* datasets that could have helped to study the contribution of the identified domain-specific characteristics. As discussed while answering RQ2 in Section 5.5.2, there are some tools that can be exploited to: i) simulate ad-hoc scenarios; ii) safely

simulate obstacles on rail tracks. Some of these, like Unreal Engine, but also Unity¹⁵ and others of this kind, would also allow for a good approximation for reality in terms of graphics and physics. In addition, they could also allow for automatic data labelling, which is often one of the most time-consuming and sensitive phases when it comes to properly characterising DL models. Then, they are extremely valuable allies to quickly and safely generate synthetic in this context. However, even though these tools can be extremely valuable for the development of PoCs, experiment strategies based on DL, and define suitable reasoning to address problems, tests with real data should be performed to validate the effectiveness of the implemented solutions.

Self-Training and Transfer Learning (↑). Together with data availability, the other data issue that was encountered concerned *data labelling*. As mentioned, some 3D Editors could automatically produce labels, nonetheless, there are cases when labels must be hand-labelled or generated from scratch. In these contexts, *Self-Training* approaches could be exploited with Transfer Learning being one of the principal boosters for this strategy. In this Thesis, this strategy has been adopted to Semi-automatic label frames extracted from RoadRunner (see Section 5.3.3). Results seem to be quite promising but, as discussed in Section 5.5, further investigation would be required to make this process more effective.

Explainability (↓), Instability (↓), Certifiability (↓), and Explainable AI (↑). To conclude, the main threats come from the *extreme aleatory of the environment*, the *explainability* and *instability* of DL models, and the *certifiability* of AI systems.

At the current stage of development of DL systems, as widely discussed in Section 2.4.1, it would be extremely challenging to statistically identify possible *failure modes* and the SIL of DL systems given, among others, their *instability* and poor *explainability*.

The application of concepts like the safety envelope (as discussed in Chapter 3) or Explainable AI approaches oriented at unrolling, i.e., explain, the behaviour of DL models, could potentially introduce

¹⁵<https://unity.com>

several opportunities towards the certifiability of systems involving DL functionalities.

5.7 Identification of Future Directions and Recommendations

To conclude with the analysis of this case study, they are here reported some general directions and recommendations that would potentially benefit future research in this context. Three main recommendations were identified, which came directly from the strengths and opportunities discussed in the previous section.

Exploit 3D Editors and Scenario Simulators. As mentioned above, 3D Editors and Scenario Simulators offer different benefits when it comes to easily and safely collecting data to develop PoCs. The latest developments give these tools high performances in approximating reality both in terms of graphic characteristics and the simulation of the physics of the elements. Their potential would then be worth to be taken into account for future research when real data would not be available or partially available.

Exploit Domain-Specific Characteristics. Although “general-purpose” DL models would be capable of performing correctly, domain-specific characteristics could be exploited to facilitate the task they should address. The example discussed in this chapter concerned exploiting the *constrained path* peculiarity to reduce the (*structural*) aleatory that the Rail Tracks Identification module, implemented through a SS approach, had to handle; hence, the model should have been “only” capable of generalising in terms of light and weather conditions, and possible intrusions.

Modular Approaches for Redundancy and Robustness. At the current stage of development, it seems that DL approaches may not be so robust to *particular* changes in the input data (i.e., are not stable, see Section 2.4.1). Some particular patterns, even imperceptible to the human eye, can induce the model to make a completely wrong

prediction. Hence, especially when it comes to safety-critical scenarios, a modular approach where each module could compensate for possible malfunctions of the others would be advisable.

Redundancy can be further improved by adding more (possibly different) sensors. A trivial example is given by the application of the proposed approach in combination with two cameras, an RGB and a thermal one. Then, the architecture in Figure 5.1 would be implemented twice, each time by taking into account data from only one of the two sensors. From a theoretical perspective, since data are different, also the resulting DL models (intended as a particular DNN trained on a specific dataset) would be different and potentially react differently to specific events. This would basically mean that, most likely, they will not fail together or, at least, that the probability of failure of the whole system would be lower than the case when only a single sensor is leveraged.

Case Study 2: Level Crossings Monitoring through Audio/Video Data

The same considerations made at the beginning of Chapter 5 hold also for this Proof-of-Concept (PoC). Then, the main objective of the analyses presented in this chapter was to investigate the potential of DL and setup a suitable strategy for *Level Crossings (LCs) continuous monitoring through non-intrusive sensors (specifically cameras and microphones)*.

6.1 Problem Statement

Level Crossings represent a class of railway assets that raise sensitive concerns in terms of safety and maintainability, especially given their role as the junction point between two domains (automotive and railways) [25]. As deductible from the yearly safety reports of the European Union Agency for Railways (the latest in 2023 [40]), they have been years now that the “railway accident picture” has not changed significantly in terms of percentages, with LCs being the stage for about 30-32% of all railway accidents. Main causes include poor visibility of incoming trains or warning signals (when present) and users’ negligence, however, also technical issues, although rare, could play a key role [25]. In addition to that, it is also worth underlining that malfunctions may also induce delays or even traffic

disruption [25]. Then, despite being safety-critical systems designed to have low failure rate, the primary step in ensuring safety and transit availability should be to assess their correct operation.

As discussed in Chapters 1 and 2, in railways, inspection activities have traditionally been conducted on the field by following a fixed schedule; LCs are not exceptions [99]. These practices, however, might not allow for timely detection of failures occurring between consecutive inspections; the shift from scheduled inspection to online monitoring would help overcome this issue. In the last years, a few solutions have been proposed to monitor both safety and maintenance LC aspects^{1,2}, collecting and analysing LC data in real-time. Besides the software, these applications owe their effectiveness especially to a series of (IoT) sensors which, as mentioned in Section 2.2.2, may introduce some complications if not included by design.

Objectives and Constraints. The principal objective would then be to understand how Level Crossings could be continuously monitored without being intrusive, i.e., adopting sensors that can be installed externally, specifically, cameras and microphones.

Existing Approaches. By analysing the existing literature, it appears that this specific combination has not received wide attention. Some AI/DL solutions have been proposed to, for example, detect faults and anomalies in LC protective barriers (e.g., [54]), however, these rely on data collected through sensors different from cameras and microphones.

Differently, cameras and DL have been analysed for anomaly/obstacle detection within the LC area for safety purposes [42], which makes cameras even more appealing as, once installed, they can be exploited to improve both LC safety and maintenance. Actually, cameras may already be installed at LCs for security purposes, hence, if those could be exploited, it might be unnecessary to make new installations.

To conclude, it is worth highlighting that the potential of DL combined with cameras and microphones (especially the former) has already been widely investigated for railway maintenance purposes (see Section 2.3.2).

¹<https://www.smartmotors.org/level-crossing-monitoring>

²<https://www.voestalpine.com/railway-systems/en/products/rxm-rail-crossing-monitoring/>

Furthermore, as already pointed out in Section 2.3.1, DL approaches have been found to be more effective and robust than traditional CV ones with respect to significant changes in the scenes to be analysed (due, e.g., to varying light and weather conditions) [125]; this is also due to the fact that the features to consider when tackling a particular task (e.g., object detection) are extracted directly by the DL model and do not have to be engineered (e.g., hand-crafted and fine-tuned) by users [125, 101]. These factors shifted the odds in favour of using DNN (presented in Section 6.2.2) to address the goal specified below.

Problem-specific Characteristics. There are basically three main characteristics that could be exploited to set up a suitable strategy for Level Crossings continuous monitoring.

First, from a high-level perspective, LCs can be classified as *active* or *passive* [72]. The latter do not have any warning signal to notify road users and pedestrians of the incoming train(s), neither have protection systems. Conversely, *active* LCs may involve all the above including: warning bells, warning lights, and protective barriers. Hence, from the outside, a LC can be seen as an integrated system composed of different warning/protective subsystems which should properly work together to ensure safety. These subsystems can be actually monitored separately.

Second, considering cameras in particular, it is not far from reality to assume that, once installed, their field of view would be fixed. This can be exploited to pre-process image data and “facilitate” the tasks that Deep Neural Networks (DNNs) have to address.

Third, it is important to underline that LCs from different countries may have different features, one of which is related to the sound of warning bells. Particularly, warning bells from different countries may have different sounds. Additionally, they could have such a particular sound that could be confused with other acoustic events recordable near LCs, like alarms, sirens, and so on.

Potential Challenges. Having stated the above, it was clear at this point that the tasks to be addressed required video data taken from a fixed camera and audio data related to warning bells, possibly from different countries to perform adequate analyses. However, after a preliminary

investigation, no data of this kind were found. Therefore, one of the primary challenges to address was related to data availability (a problem widely discussed in Section 2.4.2).

Goal. Then, the practical goals were to build datasets reflecting the aforementioned characteristics and, consequently, propose a cost-effective and non-intrusive approach for LC subsystems' continuous monitoring.

Research Questions (RQs) and Key Performance Indicators (KPIs). The goal was then formalised by means of the following RQs:

- **RQ1:** How would the above characteristics affect or support the implementation of a continuous monitoring system based on DL?
- **RQ2:** How could the problem of data availability be addressed? How can suitable datasets be generated (even synthetically) for the development of a PoC in this context?
- **RQ3:** Is it actually possible to monitor and detect anomalies in LC subsystems without using intrusive sensors? Specifically, is it possible by leveraging only cameras and microphones?

Lastly, the following KPIs were identified for the high-level evaluation of the developed approach:

- **KPI1: Implementability Costs.** Includes sensors and installation costs.
- **KPI2: Computation Time.** Refers to the time required by the system to detect possible malfunctions.
- **KPI3: Effectiveness.** Indicates performances of the approach in terms of accuracy and detectable malfunctions.

6.2 Identification of the Approach and Selection of DNNs

As mentioned above, the three LC subsystems, i.e., warning bell, warning light, and protective barrier, could be monitored separately. For the

sake of simplicity, in the following, they will *also* be referred to as “bells”, “lights”, and “barrier” respectively.

6.2.1 A Modular Approach

The modular approach was identified starting from the assumption that the functioning of the bells could have been determined by analysing audio data, while the functioning of lights and barriers could have been evaluated by leveraging video data.

Particularly focusing on lights and barriers, the idea was to first extract their behaviour over time and then compare it with the nominal one. This could have been done by exploiting one of the most known object detectors in the literature (e.g., YOLO) to simultaneously detect lights and barriers in video frames. Then, the behaviour would have been extracted as follows:

- As for lights, the detection would also have involved the classification of their status into “turned on” or “idle”. These classifications could have then been plotted over time (i.e., frame by frame) to trace the “blinking behaviour”.
- As for barriers, the heights of the bounding boxes predicted by the [DNN](#) could have been extracted. Hence, the movement of the barrier would have been reproduced by plotting these heights over time.

By taking into account both the modularity of the [LC](#) itself and the *fixed camera problem-specific characteristic*, what was indicated above was split into two tasks (warning light analysis and protective barrier analysis) and addressed through two different [DNNs](#).

The reasoning was the following: given the *fixed camera*, some pre-processing could have been applied to *statically* crop out some regions of interest (related to lights and barriers) from the frames (see Figure 6.1); hence, instead of implementing a single module to address both lights and barriers detection, two modules could have been adopted (implemented through two different [DNNs](#)). The benefit would have mainly been two-fold:

1. *Easier tasks*. Instead of having to detect multiple objects in large images, with lights being also extremely small compared to the rest of the scene, the two obtained tasks would have been much easier.

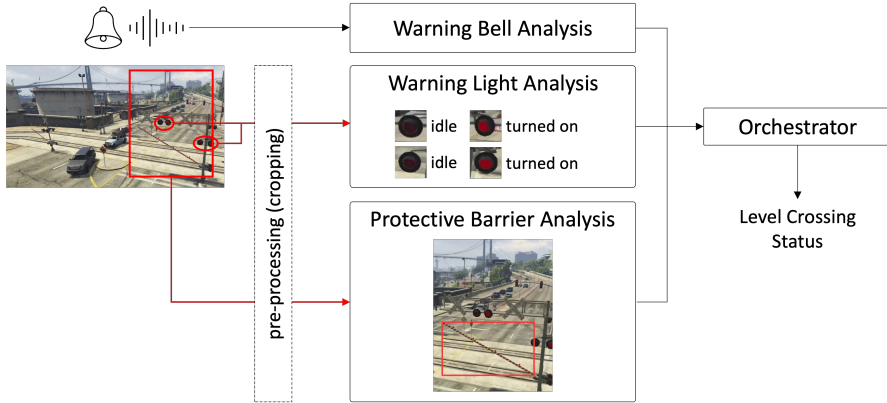


Figure 6.1. An Architecture for Level Crossing Intelligent Monitoring (Image Extracted from Grand Theft Auto V (GTA V))

As for lights, a trivial binary-classification approach could have been implemented to classify the (cropped) lights into “turned on” or “idle”. Concerning barriers, instead, the object detector would have had to focus on a single object in a smaller area compared to the whole frame.

2. *Diversity.* The adoption of a single module, in case of failure, would have compromised the analysis of both warning lights and barriers. With two modules, implemented through two different DNNs and focusing on regions of interest which are only a portion of the whole frames, that problem could have been mitigated.

On these bases, the modular approach that was identified for the implementation of the “Level Crossing Intelligent Monitoring System” (shown in Figure 6.1) includes: i) a *warning bell analysis module*; ii) a *warning light analysis module*; iii) a *barrier analysis module*; iv) an *orchestrator*. The first three modules are oriented at detecting and analysing the three corresponding components; the latter merges the outcome of the other modules and, possibly, information coming from external systems to estimate the health status of the Level Crossing.

Such a modular architecture would also have the advantage of considering Level Crossings as *black-boxes*, i.e., it will focus only on their

visible/audible behaviour and would estimate the deviations of the components from their nominal functioning. Therefore, it could be adapted to any kind of **LC**, whatever their implementation specifics are and also if not all three subsystems are present.

6.2.2 Selection of **DNNs**

Warning Bell Analysis. This module is oriented at detecting the bells. A detection problem of this kind can be formulated as an audio classification problem whose goal is to correctly identify bell sounds among all possible noises recordable near **LCs**. The most trivial solution would be to adopt a binary-classification approach to identify “bell sound activity” against “no sound activity”; however, the main problem is that this approach may lack robustness, as there are some kinds of noise that could be “confused” with bells and then wrongly classified. To mitigate this problem, a *three-class classification approach* is proposed, whose objective is to properly classify sounds belonging to the following three classes: i) *warning bell*, which represents “bell sound activity”; ii) *no alarm*, which represents “no sound activity”; and iii) *generic alarm*, which includes all sounds that could be confused with warning bells.

To that aim, it was exploited VGGish³ [64], a **CNN** named after the VGG architecture [132]. This choice was mainly influenced by the fact that, to build a suitable dataset to address this task (as discussed below), it was partially leveraged the AudioSet dataset [64] which is publicly available. VGGish has already been used to process AudioSet data, particularly, it was adopted to extract specific features from them; hence, it has already been adopted in audio-related tasks. Additionally, VGGish is deployed together with a set of pre-trained weights obtained by training it on a preliminary version of YouTube-8M³; hence, this network also allowed for the investigation the advantages introduced by Transfer Learning.

Warning Light Analysis. This module focuses on the detection of lights and the delineation of their behaviour. Two main alternatives were identified as explained above: i) adopt an object detector; ii) adopt a

³<https://github.com/tensorflow/models/tree/master/research/audioset/vggish>

simple classification network exploiting the *fixed camera* characteristic to pre-process frames. The latter seems to be more viable and easy to implement. Classification networks are typically less complex, in terms of architecture, than object detectors; also, datasets to train classification networks would be much easier to generate (especially to label).

Given the large attention that image classification has received within the literature, this module has not been experimented as potentially any kind of state-of-the-art classification DNNs could be adopted for this purpose. For the sake of completeness, considering the vision set out for this architecture, this module, for each light, would: i) classify it into “turned on” or “idle”, and ii) arrange these classifications over time to extract their “blinking behaviour”. This could then be used by the *Orchestrator* to, e.g., perform comparisons between the nominal behaviour and that detected in real-time to check the health status of the lights.

Protective Barrier Analysis. Similar to the precedent, this module aims at extracting the movement of the barrier and characterising its behaviour. Each LC has multiple barriers, however, by exploiting the *fixed camera*, the area that would be occupied by each single barrier during their entire movement can be extracted beforehand; hence, it would be possible to focus on each barrier separately.

As anticipated, an object detection approach was leveraged to extract the movement; in particular, YOLOv5⁴ was adopted. This choice was mainly driven by two factors: i) YOLO networks are among the most used networks in the literature and are also those that have generally achieved interesting performances in terms of computation times; ii) even though there could be other networks that, theoretically, could perform better than YOLO in terms of detection accuracy (e.g., Faster R-CNN), the task that is intended to be addressed in this case is much simpler than those for which these object detectors were conceived that YOLOv5 would have achieved both high detection accuracy and low computation time. The latter was actually a hypothesis, which has then been confirmed via experiments. For the sake of completeness, the problem of small-scale object detection that these networks might run into is mitigated by the pre-processing performed leveraging the *fixed camera*.

⁴<https://github.com/ultralytics/yolov5>

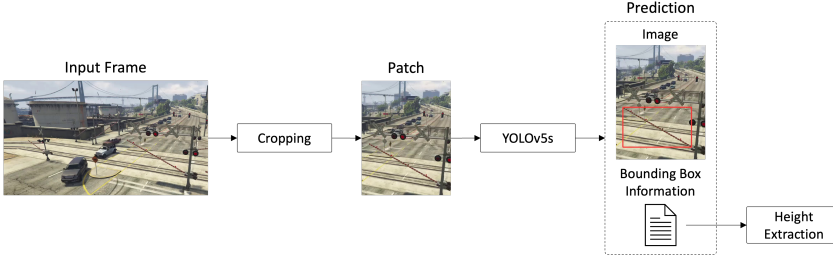


Figure 6.2. Protective Barrier Motion Extraction Pipeline

Then, as shown in Figure 6.2, the pipeline for *barrier motion extraction*, for each frame, involves the following steps: i) the smaller area containing the barrier throughout its entire movement is extracted, this area is computed beforehand and kept constant in position and dimensions; ii) YOLOv5 predicts the bounding box for the barrier; iii) the height of the bounding box is extracted. Then, the motion is identified by plotting the heights extracted from consecutive frames.

Orchestrator. As mentioned, this module would combine the outputs from the modules above and, possibly, other external information to identify the health status of the Level Crossing. The latter could include, for example, the exact moment of **LC** activation given the incoming train. This would be valuable, for example, to time the delay between the trigger of the **LC** and the moment when the components actually start functioning.

As for the *warning light analysis* module, the implementation of this module would not be detailed in this Thesis as it requires that all the above modules are implemented (possibly efficiently). However, for the sake of completeness, additional information will be given in the following to detail, as much as possible, the basic functionalities it should implement.

6.3 Identification and/or Generation of Suitable Datasets

To train and test the **DNNs** identified above, two datasets were required: an audio dataset to address audio classification and a video/frame dataset to address barrier detection.

6.3.1 Warning Bell Dataset Generation

As explained in Section 6.2.2, a three-class classification approach was identified to address the task of *warning bell detection*. Then, the dataset had to contain samples related to three classes: i) *warning bell*; ii) *no alarm*; and iii) *generic alarm*. No dataset of this kind was found to be available.

It is important to recall from Section 6.1 that bells from different countries may have different sounds. Building a general-purpose classifier that would be potentially capable of working with bells from any possible country would be extremely challenging and time-consuming as data from all bells should be collected together with possible sounds that could be “confused” with them. To resize the problem, only Italian and British bells were considered, whose sound is quite different (at least to human hearing). The double provenance has been considered to perform some in-depth analyses on the *cross-country generalisation performances* of the DNN when trained with data from a single source (e.g., Italian) and used to make predictions on data coming from the other source (e.g., British).

Having stated that, the process of dataset generation went through the following phases: *Data Collection and Labelling*, and *Data Pre-processing*.

Data Collection and Labelling. Data were collected by exploiting AudioSet and the rationale behind it. Practically, AudioSet is a collection of about two million audios (most of which have a duration of 10 seconds) which have been extracted from YouTube videos and arranged according to an ontology in about 500 classes [49]. The audios for the three classes were then collected as follows:

- *Warning bell* audios were collected by manually analysing YouTube videos. 330 10-second audio clips were collected, 165 related to Italian LCs and 165 related to British ones.
 - *No alarm* audios were collected in the same way. In this case, 350 10-second audio clips were collected which are related to background city/countryside noises (e.g., traffic noise at different intensities, wind, birds), excluding any kind of alarm (e.g., car horns).
 - *Generic alarm* audios were extracted from AudioSet. This class contains audios that are recordable near LCs and especially those
-

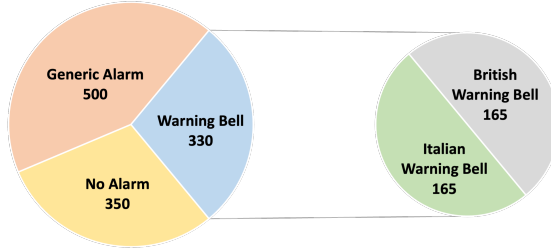


Figure 6.3. Distribution of Collected Audio Clips

that can be “confused” with warning bells.

For example, Italian warning bells have been empirically found to be similar to *other, different types of bells* (e.g., bicycle bells, church bells). British warning bells, instead, sound like *sirens* (e.g., ambulance, or emergency vehicles in general).

On these bases, different AudioSet samples were manually evaluated to select only the most relevant to this purpose. Eventually, 500 audios were extracted from AudioSet; most of which have a duration of 10 seconds. These audios belong to 19 AudioSet classes including those mentioned above.

Additional details on how this dataset was generated can be found on Zenodo⁵.

Downstream the data collection process, it was obtained an **Audio-Level Dataset** composed of 1180 audio clips, distributed as shown in Figure 6.3, most of which are 10 seconds long. These data were then randomly arranged into three sets, which will be referred to as **Audio-Level Sets**: the *Audio-Level Training Set* contains 876 audio clips ($\sim 74\%$ of the total), while the *Audio-Level Validation Sets* and the *Audio-Level Test Sets* contain 152 audio clips ($\sim 13\%$) each. Worth highlighting, the same ratio was kept for each class and “sub-class” (i.e., Italian and British bells). For example, the *Audio-Level Training Set* contains: 246 *warning bell* samples over 330 ($\sim 74\%$), where exactly half (123) is related to Italian, while the other half is related to British warning bells; 260 *no alarm* samples over 350 ($\sim 74\%$); and 370 *generic alarm* samples over 500 ($\sim 74\%$).

⁵<https://zenodo.org/records/7924875>

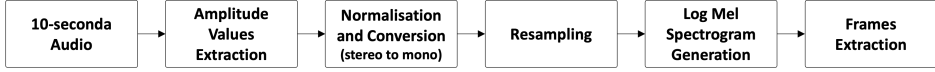


Figure 6.4. Audio Data Pre-Processing

Data Pre-processing. The CNN implemented to address this classification task shares, with the original VGGish, the feature extractor (see Section 6.4.1) and the fact that it takes in input data arranged as images. To be specific, these images are patches of a Logarithmic (Log) Mel spectrogram obtained starting from audio clips. Therefore, the audio data had to be pre-processed to obtain “images” of this kind.

Another important aspect to consider concerns the evaluation of a Transfer Learning (TL) approach since pre-trained weights were made available together with VGGish; this is the principal reason why the considered CNN and the original VGGish share the same feature extractor. To properly exploit this knowledge, the same pre-processing proposed by VGGish authors⁶ was adopted. From a high-level perspective, this pre-processing encompassed the steps depicted in Figure 6.4. Briefly, each audio was pre-processed to extract a Log Mel Spectrogram which was then portioned into K non-overlapping “frames”, where K was equal to audio’s duration in seconds (10 in most cases). Then, each *frame* represented 1 second (960 milliseconds to be specific) of the audio. Notably, in some cases, the Log Mel spectrograms were not perfectly subdivisible into exactly K frames: the last spectrogram data that would have led to “frames” smaller than 960 milliseconds were discarded.

The totality of the *frames* extracted from all audios in the Audio-Level Dataset compose the **Frame-Level Dataset** that was adopted to train and test the VGGish-inspired CNN. Important to underline, all the frames extracted from audio clips belonging to the *Audio-Level Training Set* were used to build the *Frame-Level Training Set*; the same went for the other two sets. No audio clip or frame was duplicated, hence, a specific audio (or frame) belongs to one set only.

Another aspect to consider is related to the fact that the first and last seconds of each audio may not be representative of the class due to, for

⁶<https://github.com/tensorflow/models/tree/master/research/audioset/vggish>

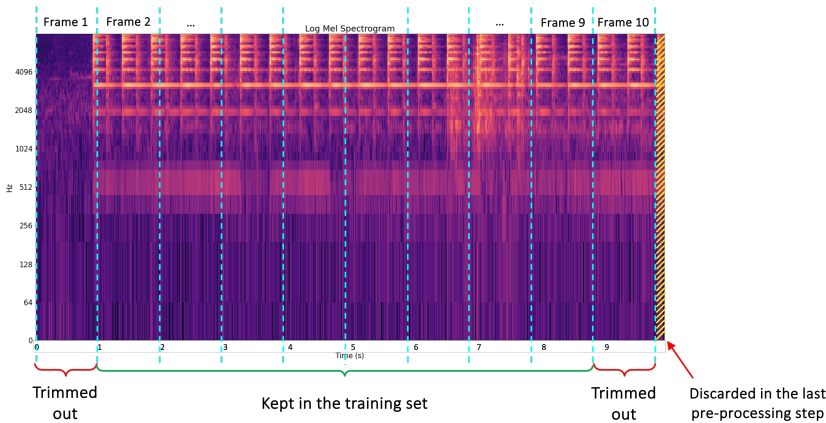


Figure 6.5. Example of Framing and Trimming Operations

example, possible noises or delays in starting or ending the recordings. Then, all the samples in the *Audio-Level Training Set* were pre-processed as specified above but, at the end, the first and the last frame of all Log Mel spectrograms were trimmed out. A graphical explanation of the framing and trimming operations is given in Figure 6.5.

To conclude, Table 6.1 reports the distribution of the samples (i.e., audio clips or frames) in training, validation, and test sets for both the *Audio-Level* and *Frame-Level* datasets: the *Frame-Level Joint Set* represents the set obtained by merging the Frame-Level Training and Validation sets, while the *Frame-Level Trimmed Training Set (TTS)* and the *Frame-Level Trimmed Joint Set (TJS)* represent, respectively, the Frame-Level Training Set and the Frame-Level Joint Set obtained after the trimming operation.

Table 6.1. Samples Distribution in Audio-Level and Frame-Level Sets

Class	Audio-Level Sets			Frame-Level Sets					
	Training	Validation	Test	Training	<i>TTS</i>	Validation	Test	Joint	<i>TJS</i>
<i>Warning bell</i>	246*	42*	42*	2460*	1968*	420*	420*	2880*	2304*
<i>No alarm</i>	260	45	45	2600	2080	450	450	3050	2440
<i>Generic alarm</i>	370	65	65	3628	2889	641	620	4269	3400
<i>Total</i>	876	152	152	8688	6937	1511	1490	10199	8144

* Exactly half of these come from Italian *bells*, the other half is related to British ones.



Figure 6.6. Example Frames extracted from [GTA](#) Videos

6.3.2 Protective Barrier Dataset Generation

As for the previous case, no suitable dataset was found. In this context, *suitable* means videos collected through a *fixed camera*, under different environmental (lights and weather) conditions, and containing the whole barrier during its movement.

Having stated that, the process of Protective Barrier Dataset generation went through the following phases: *Data Collection*, *Data Pre-processing*, *Data Labelling*, and *Data Augmentation*.

Data Collection. The first choice would have been to collect data on the field, however, this process would have not been straightforward. Therefore, for the purpose of this [PoC](#), it was leveraged a **real-life simulator** named *Grand Theft Auto V (GTA V)*⁷ (simply referred to as [GTA](#) in the following of this Thesis) to collect suitable data. [GTA](#) was born as a video game, which is still its main purpose, however, it has already been investigated to collect “real-like” data to perform experiments in contexts of, for example, autonomous cars [93]. The great advantage is that, in terms of graphics, the implemented scenarios sufficiently approximate reality; also, it allows for the easy collection of data under different light and weather conditions.

To build the dataset, a specific Level Crossing in [GTA](#) was chosen; then, **8 videos** were collected under different environmental conditions. The 8 “[GTA](#) Video Types” considered were: i) DayClear; ii) DayRain; iii) DayAfterRain; iv) DaySnow; v) NightClear; vi) NightRain; vii) NightAfterRain; viii) NightSnow. Figure 6.6 shows a frame for each of these videos.

⁷<https://www.rockstargames.com/it/gta-v>

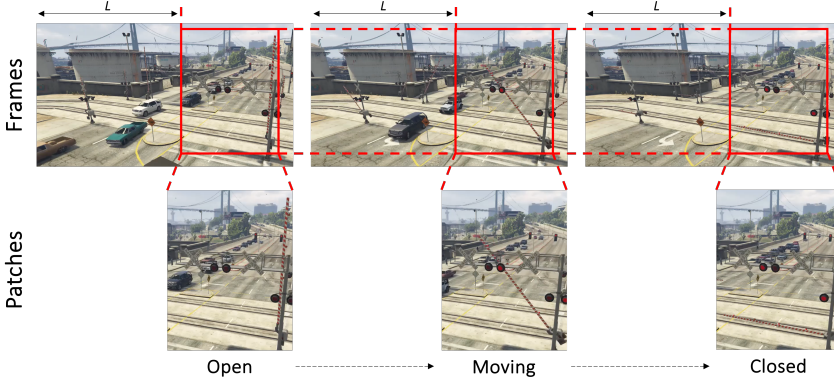


Figure 6.7. Examples of Patches Obtained from *GTA* Video Frames

Data Pre-processing. As mentioned in Section 6.2.2, *LCs* typically have more barriers; nonetheless, the *fixed camera* characteristic can be exploited to focus on each of them separately. In the context of this experimentation, attention was posed on the barrier closer to the camera.

A patch was extracted from each video frame. The position of the patch was always the same in all video frames; also, the dimensions of the patch were fixed ad-hoc to define the smallest area that would have contained the whole barrier during its movement. Figure 6.7 graphically clarify this concept. Once all frames were pre-processed, a collection of patches like those in Figure 6.7 was obtained. Worth underlining, most of the patches depicted the barrier in closed (horizontal) or open (vertical) position; frames depicting the barrier during its movement (oblique position) were much less numerous. Hence, only 240 patches per *GTA Video Type* (for a total of 1920 patches) were kept to try to balance the distribution between open, closed, and moving barriers.

Data Labelling. All 1920 patches were then manually labelled by leveraging LabelImg⁸. This is a tool that allows for the easy drawing of bounding boxes around the object of interest. Given that each patch contains a single barrier, each patch was associated with a single label identifying the position and dimensions of the bounding box containing the barrier.

⁸<https://github.com/heartexlabs/labelImg>

Data Augmentation. In order to increase diversity and the number of samples in the dataset, *Data Augmentation* was applied by leveraging the *Automold* library, similarly to what was described in Section 5.3.3. The transformations considered in this case are summarised in Table 6.2. Important to mention again, all transformations introduced some randomness, e.g., “Rain” produced drops with random intensity and angle. Hence, *there was a low probability that the same transformation applied twice on the same patch would have produced identical patches.*

To build the final Protective Barrier Dataset, patches from *GTA Video Types* were augmented by applying some of the transformations in Table 6.2. Table 6.3 shows the mapping between the transformations and *GTA Video Types*.

Notably, not all transformations were considered for each *GTA Video Type*. Some combinations (i.e., those NOT marked in Table 6.3) produced

Table 6.2. Data Augmentation Transformations (Transf.)

Transf.	Description
Bright	Randomly increase the brightness of the image
Dark	Randomly decrease the brightness of the pixel
Rain	Simulate rain by randomly adding drops with different intensities and angulation
Snow	Simulate snow by randomly increasing the brightness of specific pixels
Fog	Simulate fog by randomly blurring parts of the image
Shadow	Simulate shadows by adding random semitransparent masks on the image
SunFlare	Simulate sun flares by adding concentric and random circles in the image
Hole	Introduce noise by randomly introducing coloured ellipses within the image
Gravel	Introduce noise by simulating a random gravel pattern in the image

Table 6.3. Mapping *GTA Video Types* and Transformations

GTA Video Type	Data Augmentation Transformation								
	Bright	Dark	Rain	Snow	Fog	Shadow	SunFlare	Hole	Gravel
<i>DayClear</i>	x	x	x		x	x	x	x	x
<i>DayRain</i>	x	x	x		x	x	x	x	x
<i>DayAfterRain</i>	x	x	x		x	x	x	x	x
<i>DaySnow</i>	x	x	x	x	x	x	x	x	x
<i>NightClear</i>	x	x	x	x	x	x	x	x	x
<i>NightRain</i>	x	x	x	x	x	x	x	x	x
<i>NightAfterRain</i>	x	x	x	x	x	x	x	x	x
<i>RightSnow</i>		x	x			x	x	x	x

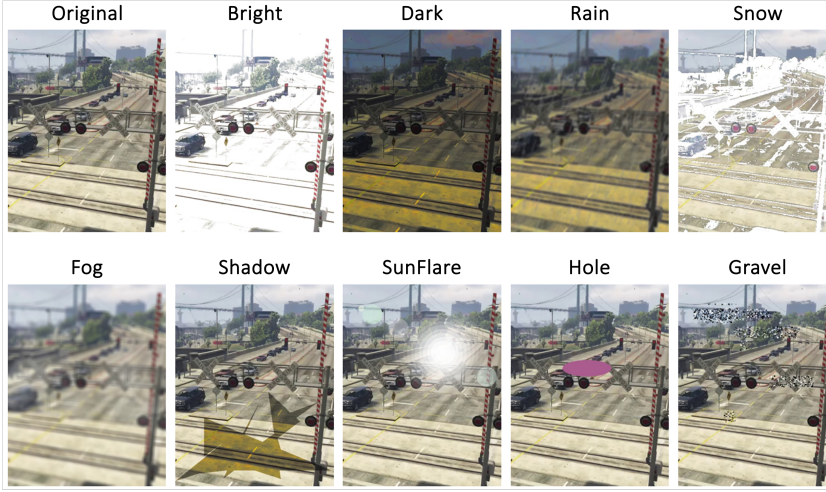


Figure 6.8. Example of Augmented (DayClear) Patches

problematic patches which were quite far from reality and/or where the barrier was not visible. Even though this phenomenon could happen in reality, if the barrier is not visible, the hand-crafted bounding box “would not have contained” any barrier, potentially affecting the detection performances of the DNN. Figure 6.8 presents the effect of the considered transformations by taking a DayClear patch as a reference. DayClear-Snow is an example of discarded combination; further details can be found in [117].

Downstream the augmentation process, there were obtained 66 combinations which, added to the 8 *GTA Video Types*, led to a dataset composed on 17760 patches (240 patches for each combination and video type). This dataset was then subdivided into Training (12423 patches, 70%), Validation (2664 patches, 15%), and Test (2664 patches, 15%) Sets. Important to underline:

- The transformations did not modify the position of the barrier within the patch, hence, the same bounding box manually computed for a given patch ‘*i*’ (of a specific *GTA Video Type*) was re-used to label all the patches that were obtained from the application of the various transformation on the patch ‘*i*’.

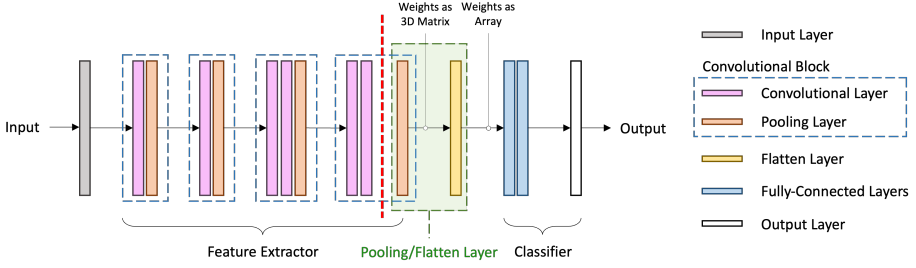


Figure 6.9. VGGish High-Level Architecture

- The ratio 70-15-15% was also kept at the combination and video type levels to make the dataset as balanced as possible. Practically, for each combination, 168 over 240 ($\sim 75\%$) patches were added to the Training Set, 36 ($\sim 15\%$) were added to the Validation Set, and the remaining 36 ($\sim 15\%$) were added to the Test Set.

In the following of this chapter, the dataset such obtained will be referred to as “**Augmented Protective Barrier Dataset**”.

6.4 DNNs Training, Validation, Test, and in-depth Experiments

This section discussed the training, validation, and test phases for the DNNs adopted to implement the Warning Bell and Protective Barrier Analysis Modules.

6.4.1 Warning Bell Analysis Module: A VGGish-inspired CNN

As mentioned in Section 6.2.2, a VGGish-inspired CNN was built to implement the Warning Bell Analysis Module. Figure 6.9 shows a high-level architecture of the original VGGish. To build the CNN for warning bell detection, the layers on the right hand of the dashed red line (i.e., the last pooling layer and the fully-convolutional ones) were dropped and substituted with other layers which were identified through the training and validation phases as discussed below.

Training and Validation

Starting from the feature extractor in Figure 6.9, which were not modified, different configurations were tested to find the one that performed the best on the *Frame-Level Dataset* (see Section 6.3.1). Specifically, all configurations were trained on the TTS and their performances were evaluated on the *Frame-Level Validation Set* (see Table 6.1).

The tests to select the best performing configuration were performed in two steps by varying and/or fixing the hyperparameters and parameters reported in Table 6.4. Particularly:

1. In the first step, the *parameters* were fixed to the values underlined in Table 6.4, while different configurations were trained and validated by varying the hyperparameters (*Hyperparameters Tuning*).
2. Then, in the second step, the best performing configurations resulting from the precedent step were taken into account. The hyperparameters were fixed while tests were performed by varying the parameters (*Parameters tuning*).

Among the hyperparameters, the *Pooling/Flatten Layer* (highlighted also in Figure 6.9) represents the combination between the last pooling operation and how weights are re-organised to be given in input to the classifier. Briefly: “glob” represents the base case proposed by VGGish’ authors, it combines a max-pooling layer (2x2 pool size and 2x2 stride)

Table 6.4. Configurations Hyperparameters and Parameters

Hyperparameters	Values
<i>Fully-Connected Layers</i>	1, 2
<i>Neurons per Fully-Connected Layer</i>	1024, 512, 256, 128 *
<i>Pooling/Flatten Layer</i>	glob, max2, max4, avg4
Parameters	Values
<i>Learning Rate</i>	<u>10^{-4}</u> , 10^{-5}
<i>Batch Size</i>	<u>20</u> , 60, 100

* and related combinations in case of 2 Fully-Connected Layers, e.g., [1024, 512].

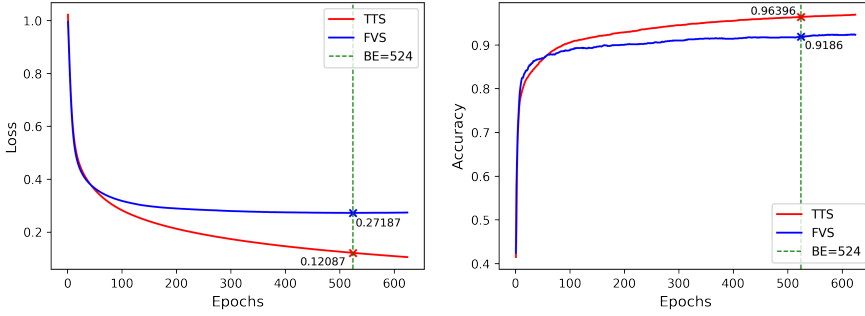
with a global avg-pooling layer; “max2” combines a max-pooling layer (2x2 pool size and 2x2 stride) with a flatten layer; similarly, “max4” and “avg4” respective combine a max-pooling layer (4x4 pool size and 4x4 stride) and an avg-pooling layer (4x4 pool size and 4x4 stride) with a flatten layer. Additional details on these and all other concepts related to the training, validation, and testing of the VGGish-inspired DNN can be retrieved by consulting references [25] (and related *supplementary data*).

All configurations were trained and validated by considering the categorical cross-entropy as the loss function (“loss” in the following), the Adam optimiser, and the categorical accuracy (“accuracy” in the following) as the evaluation metric.

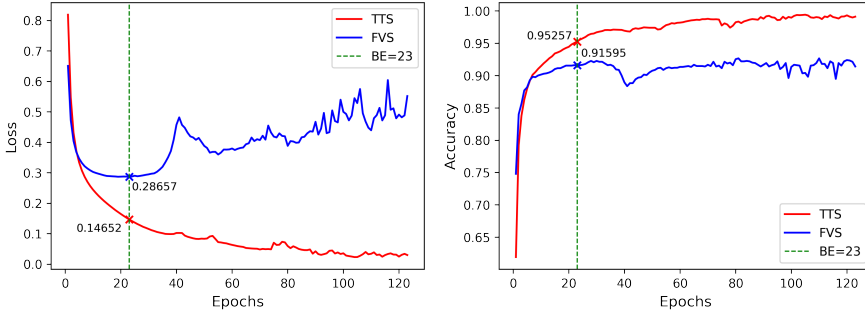
In addition to the hyperparameters and parameter in Table 6.4, there were others that were kept unchanged during all the tests: i) an Adam Epsilon (for the Adam Optimizer) equals to 10^{-8} ; ii) ReLU as the activation function of all neurons in the fully-connected layers; and iii) an output layer with three neurons and softmax as the activation function. They were considered three neurons as the *one-hot encoding* was adopted for the classes: [1,0,0] for the *warning bell* class, [0,1,0] for the *no alarm* class, and [0,0,1] for the *generic alarm* class.

To conclude, as identified in Section 2.1.2, the training can be performed in two main ways: *from scratch* and by *leveraging TL*. In this case, all configurations were trained by loading and freezing the pre-trained weights of the convolutional blocks which were made available by VGGish authors. Hence, during the training, only the weights of the fully-connected layers were updated. This choice came from some preliminary tests that highlighted the advantages of this strategy compared to loading weights but not freezing them or training the CNN from scratch; details on the benefits that TL brought in this context are given below.

Hyperparameters Tuning. A total of 56 configurations were evaluated; they were obtained by varying the hyperparameters and fixing the learning rate and the batch size to 10^{-4} and 20 respectively. Each configuration was trained for a maximum of 2000 epochs and the *Early Stopping* criterion was adopted so that the training would have stopped if the *loss computed on the Frame-Level Validation Set* (referred to as *validation loss* in the following) did not improve for 100 consecutive epochs.



(a) Stable Configuration: avg4, one fully-connected layer with 128 neurons.
FVS: Frame-Level Validation Set – BE: Best Epoch.



(b) Unstable Configuration (with evident oscillations): max2, two fully-connected layers with 512 and 128 neurons respectively.

Figure 6.10. Examples of stable and unstable configurations

The first screening was performed by discarding all configurations that: i) *evidently diverged*, ii) *were characterised by heavy oscillations* (i.e., were not ‘stable’⁹). Figure 6.10 graphically shows the concepts of ‘stable’ and ‘unstable’ configuration. All configurations that were compliant with these characteristics were discarded. The others, instead, after a specific training epoch, reached a sort of plateau in terms of validation loss so that all of them could have been considered equivalent in terms of stability. Hence, an empirical threshold was set and all configurations that had a validation loss

⁹In this context, ‘stable’ means that, starting from a specific epoch (which varies according to the configuration), the validation loss does not significantly vary over time.

greater than 0.28 were discarded as well. Table 6.5 reports the performances of the survived configurations evaluated at the best epoch (i.e., the epoch at which they achieved the lowest validation loss).

Parameters Tuning. The configurations that passed the screening above were then further analysed by keeping constant their hyperparameters and varying the parameters. In this phase, 16 additional configurations were considered. For these tests, the number of maximum epochs and the *Early Stopping* patience respectively rose to 20000 and 300.

The set of configurations achieving the best performances resulted to be those having a *single fully-connected layer with 1024 neurons, avg4 as the Pooling/Flatten Layer*, and a *learning rate equals to 10^{-5}* . As can be noticed from Table 6.6, the performances resulted to be quite independent from the batch size. However, to perform additional tests on the viability of this network, it was selected the configuration that achieved the lowest validation loss (i.e., that with *batch size equals to 20*); this resulted to be also the one with the highest accuracy on the Frame-Level Validation Set. This is referred to as the “best performing configuration” and its training curves are shown, for the sake of completeness, in Figure 6.11.

Network Testing and In-depth Experiments

Starting from the best configuration, some experiments were performed to evaluate the following aspects:

Table 6.5. Best Configurations after Hyperparameters Tuning

P/F Layer	FC Layers	Neurons per FC Layer	Validation Loss	Best Epoch	Validation Accuracy	Training Accuracy	Training Loss
avg4	1	128	0.271869	524	0.918596	0.963961	0.120873
glob	1	1024	0.276927	640	0.915287	0.942194	0.171911
glob	2	1024, 256	0.277950	266	0.917935	0.941761	0.165159
glob	1	256	0.278084	1241	0.915287	0.942049	0.169504
glob	1	512	0.278849	833	0.914626	0.940320	0.176325
avg4	1	256	0.279265	328	0.917935	0.961366	0.131476
glob	2	256, 128	0.279277	317	0.914626	0.938734	0.180421
avg4	1	1024	0.279867	144	0.910655	0.949834	0.158503

P/F: Pooling/Flatten – FC: Fully-Connected.

Validation Loss/Accuracy: Loss/Accuracy evaluated on the Frame-Level Validation Set.

Training Loss/Accuracy: Loss/Accuracy evaluated on the TTS.

Table 6.6. Best Configurations after Parameters Tuning
P/F Layer: *avg4* – *Single* FC Layer with *1024* Neurons

Learning Rate	Batch Size	Best Epoch	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1e-5	20	1663	0.968718	0.110086	0.922567	0.268660
	60	2692	0.966267	0.114772	0.919258	0.269418
	100	3163	0.964826	0.119249	0.922567	0.269779

P/F: Pooling/Flatten – FC: Fully-Connected.

Validation Loss/Accuracy: Loss/Accuracy evaluated on the Frame-Level Validation Set.

Training Loss/Accuracy: Loss/Accuracy evaluated on the **TTS**.

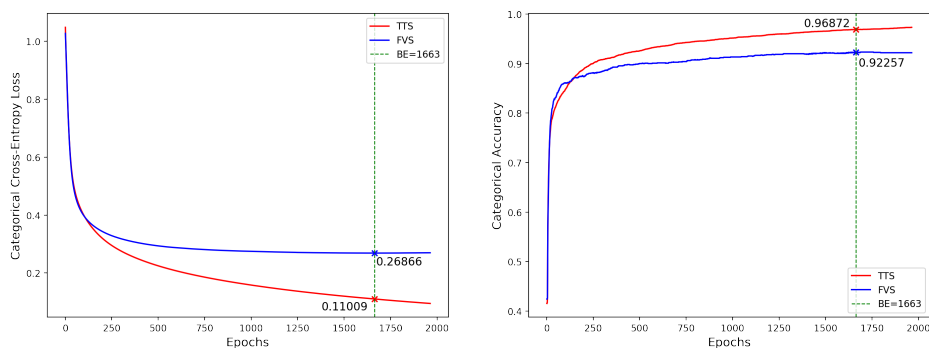


Figure 6.11. Training Curves of the Best Performing Configuration

- Assess the advantages that **TL** brings in case only a limited amount of data is available.
- Evaluate how the best configuration performs on test data, both at the Frame-Level and at the Audio-Level.
- Analyse how the performance of the network varies with the size of the training set (i.e., the **TTS**).
- Evaluate the *cross-country* generalisation performances.
- Assess the robustness of the network to environmental noises.

Worth highlighting, as also for the following tests, additional details can be visioned by accessing references [25, 117].

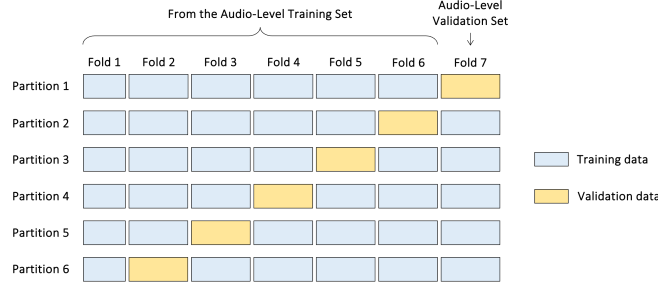


Figure 6.12. Cross-Validation Folds and Partitions

Advantages of Transfer Learning. As mentioned, all configurations were trained by loading and freezing the pre-trained weights of the feature extractor. Here is shown the contribution that TL bring in this case by taking the best configuration as a reference.

An analysis based on the cross-validation (see Section 4.2.4) was performed to evaluate the performance of the best configuration by varying the *weights composition*:

1. Pre-trained weights *Loaded And Frozen*; as discussed so far.
2. Pre-trained weights *Loaded but Not Frozen*; practically, they are used only to *initialise* the weights of the convolutional blocks, then, and are then left free to evolve during the training.
3. Pre-trained weights *Not Loaded*; i.e., training from scratch.

To perform the cross-validation, data had to be subdivided into *folds*. Then, the Audio-Level Validation Set was considered as a fold and, consequently, the Audio-Level Training Set was partitioned into 6 folds such that each of the 7 folds would have contained about the same data distribution in terms of number of samples, i.e.: 42 *warning bell* audios (21 from Italian and 21 from British LCs), 45 *no alarm* audios, and 65 *generic alarm* audios. Notably, this is actually the composition of the Audio-Level Validation Set (see Table 6.1). One of the fold had a lower number of data compared to the others because the number of samples (class-by-class) in the Audio-Level Training Set was not perfectly subdividable by 6; this specific fold was left aside and its data were used only as training data as show in Figure 6.12.

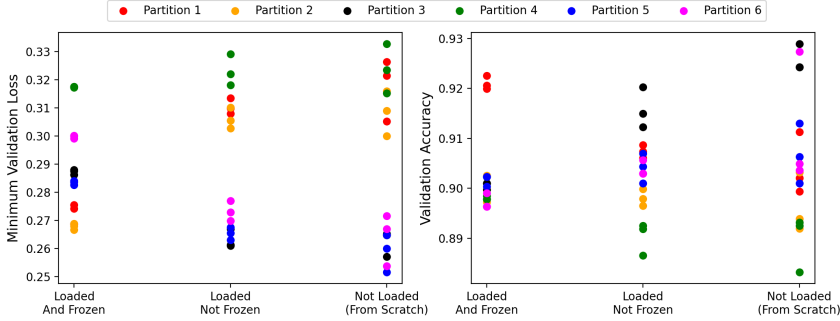


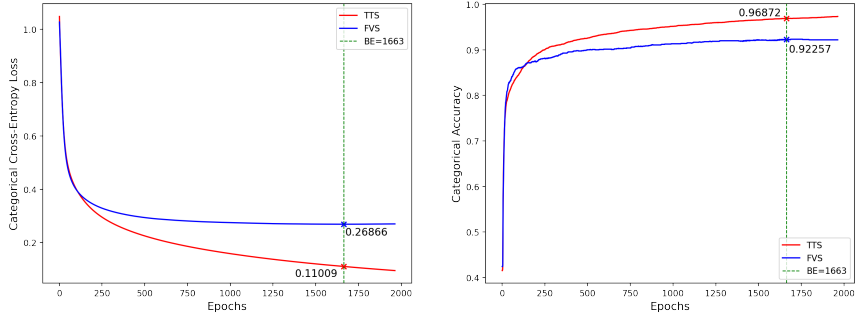
Figure 6.13. Cross-Validation Results

Training and validation data, for each partition, were then pre-processed as described in Section 6.3.1. Notably, *Partition 1* leads to the TTS and Frame-Level Validation Set discussed above.

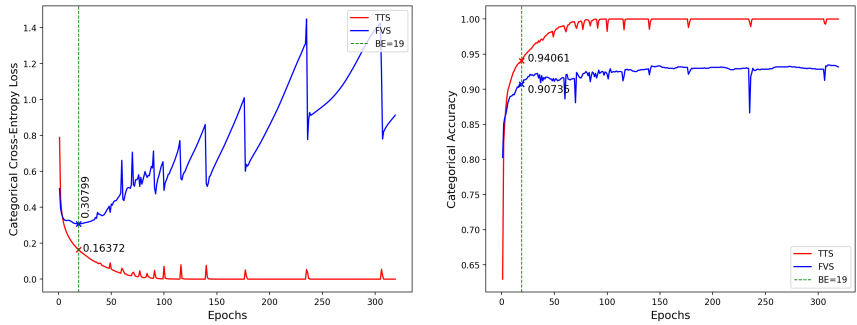
The best configuration was re-trained multiple times, each time by varying the partition (i.e., training and validation data) and weights composition. Actually, the training was repeated three times for each combination (partition - weights composition), in order to make more accurate analyses.

Figure 6.13 shows the results of these analyses in terms of Minimum Validation Loss and Validation Accuracy achieved in each training experiment. Conceptually, if a model is *stable*, when re-trained multiple times under the same conditions, it should achieve about the same performances: this is what happens in case the weights are loaded and frozen, where, especially looking at the minimum validation losses, points of the same colour are extremely close or even overlap. Furthermore, by analysing the validation accuracies, in case weights are loaded and frozen most of accuracy values set within a small range. Conversely, in case weights are *loaded but not frozen* or *not loaded*, the performances in terms of both losses and accuracies spread over larger intervals. For the sake of completeness, in these two cases, there is also sharp separation in terms of loss values; Partitions 3, 5, and 6 can then be considered as the “optimal” ones (see Section 4.2.4) as they lead to much lower loss values compared to those achieved on the other (“non-optimal”) partitions.

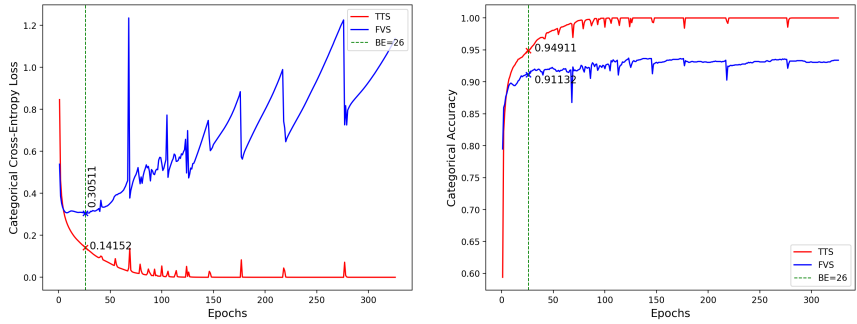
Then, Figure 6.14 shows the training curves related to three of the performed experiments on *Partition 1*. These three have been chosen for



(a) Weights Loaded and Frozen (as in Figure 6.11)
FVS: Frame-Level Validation Set – BE: Best Epoch.



(b) Weights Loaded but Not Frozen.



(c) Weights Not Loaded (Training From Scratch).

Figure 6.14. Training Performances of the Best Configuration on *Partition 1* under Different Weights Compositions.

demonstrative purpose only, nonetheless, it is extremely worth underlining that all other training curves, independently from the specific partition, had the same trend, i.e.: in case weights are loaded but not frozen (Figure 6.14b) or are not loaded (Figure 6.14c), the curves are characterised by some (heavy) oscillations. Even though in these cases the configuration achieves lowest loss values and/or higher accuracy under some circumstances, by looking at the picture as a whole, the results from these tests seem to validate the theoretical hypothesis that TL offers a suitable trade-off between performances and network stability even if the dataset is not optimal or large enough, as in this case, to properly characterise a DNN.

Worth highlighting, these analyses were performed with the aim of showing how cross-validation could be applied. When implementing a DNN to be practically adopted for operational purposes, analyses of this kind should be considered also when performing other tests (e.g., parameters and hyperparameters tuning) to select the configuration that, statistically, would achieve an optimal trade-off between accuracy and stability.

To conclude, given the results above, the following tests have then been performed by considering the best configuration trained by loading and freezing pre-trained weights.

Performances on Test Sets. This test was oriented at evaluating how the best configuration would have performed on data it had never seen before, i.e., on the Frame-Level and the Audio-Level Test Sets.

The best configuration was first re-trained by loading and freezing the pre-trained weights on the TJS (see Section 6.3.1 and Table 6.1) for 1663 epochs, i.e., those related to the lowest validation loss during the training phase (first line in Table 6.6).

Table 6.7 shows the confusion matrices at both the Frame-Level and the Audio-Level, while Table 6.8 reports the evaluation metrics (Precision, Recall, and F1-score) evaluated on the test sets at both levels and for each class. Conceptually, considering a generic class X as a reference:

- *Precision* indicates how many of the samples classified as X actually belong to that class. It is computed as:

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

where “True positives” represents the number of samples correctly classified as X , while “False positives” represents the number of samples wrongly classified as X .

- *Recall* indicates how many of the samples actually belonging to the class X are correctly classified as X . It is computed as:

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

where “False negatives” represents the number of samples belonging to the class X which are not classified as X .

- *F1-score* combines Precision and Recall into a single value as follows:

$$F1\text{-score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Table 6.7. Confusion Matrices at the Frame-Level and the Audio-Level

		Prediction		
		WB	NA	GE
Real Class	WB	388	10	22
	NA	11	417	22
	GE	23	24	573
		Frame Level		

		Prediction		
		WB	NA	GE
Real Class	WB	42	0	0
	NA	0	43	2
	GE	0	2	63
		Audio Level		

Table 6.8. Precision, Recall, and F1-score per Class

Class	Frame Level			Audio Level		
	Precision	Recall	F1-score	Precision	Recall	F1-score
WB	0.9194	0.9238	0.9216	1	1	1
NA	0.9246	0.9267	0.9256	0.9556	0.9556	0.9556
GE	0.9287	0.9242	0.9264	0.9692	0.9692	0.9692

As a general rule, the closer these measures are to 1, the higher the performance of the network.

As for the accuracy, the CNN, once re-trained on the TJS for 1663 epochs, achieved an accuracy of 92.48% on the Frame-Level Test Set, while the accuracy on the Audio-Level Test Set was equal to 97.37%

For the sake of clarity, there are two aspects that require to be detailed:

- To understand whether an audio clip was correctly classified, considered that it can be partitioned into K frames when pre-processed (see Section 6.3.1), it was possible to average the K Frame-Level predictions to generate a single Audio-Level prediction. Hence, to compute the Audio-Level accuracy, they were compared the Audio-Level predictions such obtained with the actual classes of the audio clips.
- The Frame-Level Test Set actually contains the same information as the Audio-Level Test Set, but data are arranged in frames of about 1 second rather than in 10-second audio clips (actually spectrograms). Being the “same” data, the Audio-Level accuracy is higher than the Frame-Level accuracy simply because possible miss-classifications at the Frame-Level are somehow absorbed when computing the average to obtain the Audio-Level prediction.

To conclude, it was also evaluated the *time computation* performance of the CNN. It takes, on average, ~ 1.13 ms to process a single 1-second frame.

Performance Variation against Training Set Dimension. The analyses on TL have shown that the collected dataset seemed not to be large enough to properly characterise the selected CNN if TL was not applied. Additional tests were performed to evaluate, in detail, the correlation that holds between the dataset’s dimensions and the network’s performances.

The best configuration was then re-trained multiple times (always by loading and freezing pre-trained weights) by considering training sets of different dimensions. Particularly, starting from the TJS, its dimensions were reduced by 5%, 30%, 45%, and 60%. Hence, the performances were evaluated on the Frame-Level Test Set, which remained unchanged and

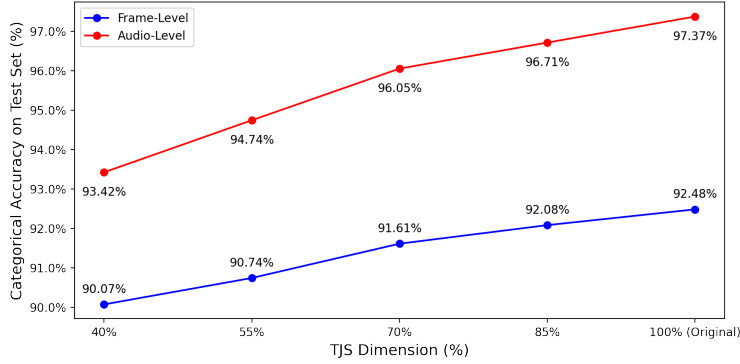


Figure 6.15. Frame-Level and Audio-Level Accuracy on Test Sets per Reduced TJS

was used as a reference to compare results. Important to mention, the reduction of the TJS was performed class-wise: practically, considering the “85% TJS”, each class or sub-class (i.e., Italian *warning bell*, British *warning bell*, *no alarm*, and *generic alarm*) were reduced by 15% to keep the resulting training set as balanced as possible.

Figure 6.15 graphically shows the accuracy trends (evaluated on the test sets) in relation to the dimension of the TJS. Notably, the larger the training set, the better the performances. These results seem to validate the hypothesis that there would still be room for improvements if additional training data were collected.

Cross-Country Generalisation Performances. As stated in Section 6.1, warning bells from different countries may have different sounds. To evaluate how this aspect could affect the cross-country generalisation performances of the best configuration, two tests were performed:

- First, the CNN was trained with Italian *warning bells* only, and evaluated on the whole Frame-Level and Audio-Level Test Sets to understand how it would have classified, especially, British *warning bells*. The same test was then performed again by training the CNN with British *warning bells* only.
- Second, 60 additional audio clips were collected: 30 from German LCs

and 30 from French LCs. After pre-processing, they were obtained 300 frames for each of these “sub-classes”. The test was then oriented at evaluating the cross-country classification performance of the CNN (when trained on the whole TJS) on these new frames.

For the first test, they were generated two “*One-country*” training sets starting from the TJS. The “*Italian only TJS*” contains all Italian *warning bell* frames that were in the TJS; then, given that Italian *warning bell* frames are exactly half of all *warning bell* frames in the TJS, also the other classes were reduced by half by randomly discarding half of the samples. The other way around was done to generate the “*British only TJS*”.

Tables 6.9 and 6.10 analyse in detail how the Italian and British *warning bells* in the Frame-Level Test Set were classified when the network was trained on “*Italian only TJS*” and “*British only TJS*” respectively. As expected, the CNN showed *poor cross-country generalisation performances*. This is most likely due to the fact that Italian and British bells are quite different, hence, the CNN is not be capable of learning how to correctly classify British bells if only Italian bells are included in the training set. Indeed: in the case of “*Italian only TJS*”, 181 over 210 (86.19%) British

Table 6.9. Classification Performance on Warning Bell Frames in Test Set when Trained on “*Italian only TJS*”

WB Sub-class	Tot. Frames in Test Set	Correctly Classified	Classified as NA	Classified as GE	Total Misclassified
<i>Italian</i>	210	193	4	13	17
<i>British</i>	210	29	25	156	181

WB: Warning Bell | NA: No Alarm | GE: Generic Alarm

Table 6.10. Classification Performance on Warning Bell Frames in Test Set when Trained on “*British only TJS*”

WB Sub-class	Tot. Frames in Test Set	Correctly Classified	Classified as NA	Classified as GE	Total Misclassified
<i>Italian</i>	210	26	10	174	184
<i>British</i>	210	195	6	9	15

WB: Warning Bell | NA: No Alarm | GE: Generic Alarm

warning bell frames were misclassified; similarly, in the case of “British only TJS”, 184 over 210 (87.62%) Italian *warning bell* frames were misclassified. Consequently, about the same happened at the Audio-Level:

- In the case of “Italian only TJS”: 18 British *warning bell* audio clips over 21 in the Audio-Level Test Set were misclassified (2 as *no alarm* and 16 as *generic alarm*).
- In the case of “British only TJS”: 18 Italian *warning bell* audio clips over 21 in the Audio-Level Test Set were misclassified (all as *generic alarm*).

To further assess cross-country generalisation performances, the second test introduced above was performed. Table 6.11 shows the classification performances of the CNN (trained on the whole TJS) on German and French *warning bell* frames:

- Concerning German Warning Bells: only 6.33% accuracy was achieved at the Frame-Level, while only a single audio clip over 30 (3.33% Audio-Level accuracy) was correctly classified.
- As for French Warning Bells: the CNN performed slightly better by achieving 33% accuracy at the Frame-Level and correctly classifying 9 audio clips over 30 (30% Audio-Level accuracy).

Despite the fact that these tests might not be comprehensive enough to generalise findings, the results discussed above seems to show that a general-purpose classifier would be extremely challenging to build as numerous sample from any possible *warning bell* source (e.g., country) should be

Table 6.11. Classification Performance on German and French *warning bells* when Trained on TJS

WB Country	Total Frames	Correctly Classified	Classified as NA	Classified as GE	Total Misclassified
<i>German</i>	300	19	11	270	281
<i>French</i>	300	99	10	191	201

WB: Warning Bell | NA: No Alarm | GE: Generic Alarm

collected. Therefore, it would be advisable to build an ad-hoc classifier for each *warning bell* source by “simply” keeping the same CNN architecture but varying the training set.

Robustness to Noise. As the last tests, the robustness of the CNN – trained on the TJS by loading and freezing the pre-trained weights – has been evaluated against three *Noise Types*: i) *Gaussian Noise*, synthetically generated; ii) *Ambulance Noise*, real noise related to the siren of an ambulance; and iii) *Motorcycle Noise*, real noise related to the engine of a motorcycle. The audio clips representing ambulance and motorcycle noises (one audio for each of them) were not included in any of the Audio-Level Sets. Noises were taken into account separately and added to each audio clip in the Audio-Level Test Set; hence, for example, the Gaussian-Augmented Test Set contains the original audio clips of the Audio-Level Test Set each of which has been modified by summing the Gaussian noise to it.

Actually, CNN performances were also evaluated at different *noise “intensities”* represented by the *Signal-to-Noise Ratio (SNR)* which is computed as reported in Equation 6.1.

$$SNR = \frac{RMS_{Audio}^2}{RMS_{Noise}^2} \quad (6.1)$$

where RMS stands for Root Mean Squared value. By fixing the SNR and knowing the RMS_{Audio} of each audio clip in the Audio-Level Test Set, it could have been retrieved the specific $RMS_{Required}$ (by applying the inverse formula) that the noise must have before being added to each audio clip to generate a noisy audio whose SNR is equal to the desired one.

On these bases, 21 *Audio-Level Noisy Test Sets* were generated by combining *Noise Types* with seven pre-specified SNR Levels. Then, by applying the same pre-processing described in Section 6.3.1, they were obtained the Frame-Level Noisy Test Sets.

Table 6.12 reports the performance of the CNN evaluated on each *Noisy Test Set* in terms of accuracy, and Recall and Precision evaluated only for the *warning bell* class; the objective was to understand how the noises (at different SNR levels) would have affected CNN performances in the classification of *warning bell* samples. In the table, each *Noisy Test Set* is represented by the combination of the *Noise Type* and the SNR Level.

As a general rule, **CNN** performances decrease with the increment of the **SNR** Level; i.e., the higher the contribution of the noise, the worse the classification performances. However, interestingly, in the case of *Ambulance Noise*, the *WB Recall* remains higher than 0.9 at the Frame-Level (independently from the **SNR** Level), while is equal to 1 up to an **SNR** Level of 2.5. Similarly, the *WB Precision* sets around 0.8 at the Frame-Level, while is greater or equal to 0.9 at the Audio-Level. These results indicate that the **CNN** is quite robust to the *Ambulance Noise* as most of (or all) *warning bell* samples were classified correctly (*WB Recall*), and that there is a low miss-classification rate for *warning bell* samples (*WB Precision*). On the other hand, in the cases of *Gaussian* and *Motorcycle Noises*, despite the *WB Precision* being quite high (up to a **SNR** of 1), the **CNN** performances in terms of accuracy and *WB Recall* degrades quickly with the increment of **SNR** Level. This basically indicates that the higher the contribution of the noise, the more the **CNN** struggles to correctly detect *warning bells* samples.

Table 6.12. Classification Performances on Noisy Test Sets

Noisy Test Set		Frame Level			Audio Level		
Noise Type	SNR Level	Accuracy	WB Precision	WB Recall	Accuracy	WB Precision	WB Recall
Gaussian	50	88.39%	0.9735	0.7881	95.39%	1.0000	0.8810
	25	85.97%	0.9870	0.7238	93.42%	1.0000	0.8333
	10	80.20%	0.9915	0.5548	84.87%	1.0000	0.6190
	5	73.36%	1.0000	0.3738	78.29%	1.0000	0.4286
	2.5	66.17%	1.0000	0.1667	69.08%	1.0000	0.1429
	1	57.58%	1.0000	0.0286	60.53%	1.0000	0.0238
	0.1	43.56%	0.0000	0.0000	40.79%	0.0000	0.0000
Motorcycle	50	87.18%	0.8606	0.8500	94.08%	0.9767	0.9524
	25	83.96%	0.8340	0.8238	90.13%	0.8936	0.9048
	10	77.85%	0.8057	0.7452	80.26%	0.8571	0.8095
	5	71.21%	0.7912	0.6405	73.68%	0.8077	0.6429
	2.5	65.17%	0.7700	0.5310	69.08%	0.7778	0.5476
	1	57.05%	0.7602	0.3810	56.58%	0.8000	0.3571
	0.1	43.22%	0.7750	0.0548	42.76%	0.8621	0.0000
Ambulance	50	89.87%	0.8541	0.9405	96.71%	1.0000	1.0000
	25	88.46%	0.8219	0.9452	94.74%	0.9744	1.0000
	10	86.64%	0.7964	0.9476	92.76%	0.9189	1.0000
	5	85.57%	0.7642	0.9381	90.13%	0.9000	1.0000
	2.5	84.03%	0.7611	0.9167	88.16%	0.9583	1.0000
	1	83.29%	0.7477	0.8833	87.50%	0.9375	0.9524
	0.1	68.86%	0.5111	0.5905	71.05%	0.0000	0.5952

WB: Warning Bell

6.4.2 Protective Barrier Analysis Module: YOLOv5

YOLOv5 comes in five configurations: nano (n), small (s), medium (m), large (l), and Xlarge (x). As explained by the authors:

*“larger models like YOLOv5x [...] will produce better results in nearly all cases, but have more parameters, require more CUDA memory [i.e., more GPU RAM] to train, and are slower to run. For mobile deployments we recommend YOLOv5s/m, for cloud deployments we recommend YOLOv5l/x”*¹⁰.

YOLOv5 authors evaluated the mean Average Precision (mAP) and computation time that these configurations achieved on the MS COCO 2017 dataset¹¹, which contains about 80 different objects. As a general rule, the bigger the model, the higher the mAP; on the other hand, the smaller the model, the lower the computation time to process a single image¹².

The selection of the YOLOv5 model, together with the strategy to train and validate it, were based on the following hypothesis:

Given:

- The advantages introduced by the exploitation of the *fixed camera*, that allowed to crop the initial frame to obtain patches containing a single barrier which, also, occupies a large portion of it (no small-scale detection problem).
- The specific goal, which can be conceptually defined as “single-object detection” and, hence, is extremely simpler than multi-object detection tasks YOLOv5 has been developed for and tested against.

Then:

- One of the smaller models could have achieved high performances both in terms of detection accuracy and computation time, hence, *YOLOv5s* was considered to address the barrier detection task.
- Conversely from what happened for VGGish that had to be *re-adapted* (Section 6.4.1), YOLOv5s could have been re-used

¹⁰<https://github.com/ultralytics/yolov5/wiki/Tips-for-Best-Training-Results>

¹¹<https://cocodataset.org/#home>

¹²<https://github.com/ultralytics/yolov5>

without hard hyperparameters/parameters tuning and would still have achieved high performance.

Hence, YOLOv5s was trained and validated by leveraging the parameters that YOLOv5s authors found to be the best in their tests on the MS COCO dataset. Similarly, they used the pre-trained weights (i.e., [TL](#) was applied also in this case) that YOLOv5s authors obtained by training YOLOv5s on the COCO dataset. Then, in addition to the augmentations performed in Section 6.3.2, the augmentations proposed together with YOLOv5 (e.g., mosaic¹³) were adopted to further increase training set samples. The contributions of pre-trained weights and additional transformations in YOLOv5s detection performances are analysed below when testing the network.

Training and Validation

YOLOv5s was trained on the *Augmented Protective Barrier Training Set* and evaluated on the *Augmented Protective Barrier Validation Set*. To meet memory constraints, i.e., not to exceed GPU ram during the training, the *batch size* was set to 128, while the images were resized (*image size* equals to 320 pixels) before being presented in input to the network. Then, the *maximum number of epochs* was set to 300.

YOLOv5 was trained by considering its own *composite loss function*, referred to as **Total Loss** in the following). It is given by the weighted sum of three contributions¹³: i) *Classes Loss*, that evaluates the error the network produces in classifying the detected object (in this specific case, given the sing-object detection, this error was always equal to 0); ii) *Objectness Loss*, that evaluates the error the network makes in detecting whether an object is present in a grid cell; and iii) *Location Loss*, represents the error made in localising an object within a grid cell. Then, two different *optimizers*, Stochastic Gradient Descent and Adam, were considered to understand which of them would have led to the best performances. Lastly, an *Early Stopping* criterion with a patience of 60 was considered to stop the training. The *Early Stopping* was used in combination with a specific function, named *Fitness Function*, which is a linear combination of some specific [mAP](#) metrics:

¹³https://docs.ultralytics.com/yolov5/tutorials/architecture_description

$$FitnessFunction = 0.1 * mAP@.5 + 0.9 * mAP@.5 : .95$$

The training was then oriented at finding the configuration of the weights that would have maximised the Fitness Function.

For the sake of completeness, it is worth mentioning that the *mAP* is essentially the mean of the Average Precision scores computed for each object to be detected (contained in the dataset); the Average Precision is computed as the area under the Precision-Recall curve. In this case, where only a single object has to be detected (i.e., the barrier), the *mAP* and the AP coincide; however, the *mAP* annotation is still used for consistency.

From a conceptual perspective, Precision is a measure that indicates how many of the predicted bounding boxes actually contain a barrier; while Recall indicates how many barriers (contained in the dataset) have been correctly detected. Precision and Recall are computed as already shown in Section 6.4.1, nonetheless, when it comes to object detection, the identification of True positives, False positives, and False negatives is made by considering an Intersection over Union (*IoU*) threshold as a reference. For example, considering an *IoU* threshold of 0.5: if the predicted bounding box overlaps for at least 50% with the label, the detection is “correct” and accounts for a True positive; if the two does not overlap, or overlap for less than 50%, the prediction is considered as a False positive; lastly, if no bounding box is predicted but the image contains a barrier, a False negative occurs. Having state that: *mAP@.5* means that the *mAP* has been computed by considering an *IoU* threshold of 0.5; while, *mAP@.5:.95* is given by the mean of the ten Average Precision scores computed with different *IoU* thresholds (from 0.5 to 0.95 with a step of 0.05).

Figure 6.16 shows the training curves – computed on the Augmented Protection Barrier Validation Set – of YOLOv5s when trained by considering Stochastic Gradient Descent or Adam as the *optimizer*. Notably, the *mAP@.75* was considered instead of *mAP@.5* as the former, as a metric, is more strict than the latter. The aim of the implemented approach was to extract barrier motion starting from bounding boxes’ heights; hence, bounding boxes must be properly predicted to extract adequate motion features. *mAP@.5* can be adopted to generally evaluate if an object detector performs properly, however, even though the *mAP@.5* is high, it is not said that the predicted bounding box is accurate enough: as said, they must

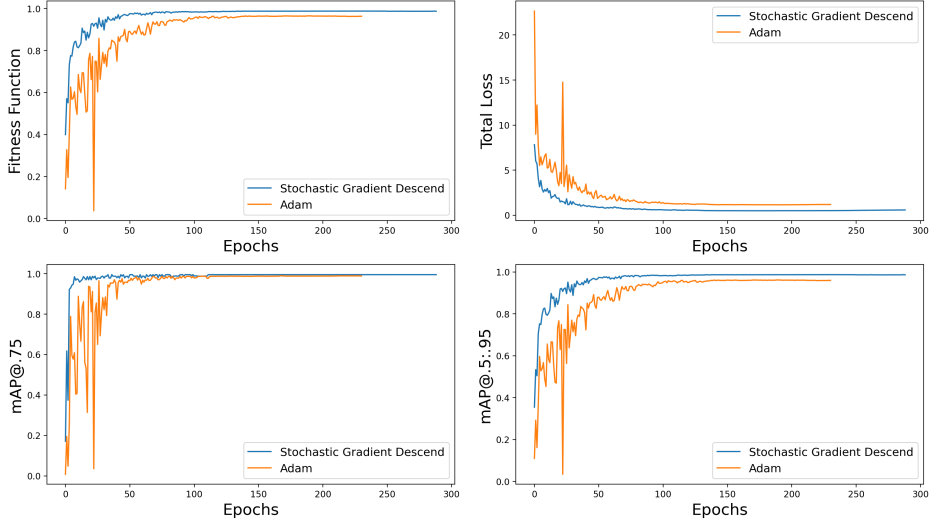


Figure 6.16. YOLOv5s Performances on Validation Set in case of Stochastic Gradient Descent and Adam Optimizer

overlap for 50% “only” with the ground truth to be considered correct). Therefore, $\text{mAP}@.75$ was considered to be more appropriate to evaluate network performances as, to be high, the predicted bounding boxes must overlap for a higher percentage (75%) with the ground truth. Having clarified that, the results in Figure 6.16 shows that, in the case of Adam, training curves are characterised by heavy oscillations during the first epochs; then, they stabilise towards the end but, overall, they are slightly worse than those obtained when using the Stochastic Gradient Descent.

For the sake of completeness, some other tests were performed by varying the composition of the Fitness Function, specifically, by adding a third component based on the $\text{mAP}@.75$. YOLOv5s was then trained multiple times (always considering the Stochastic Gradient Descent) by varying the weights of the new Fitness Function:

$$\text{FitnessFunction} = w_1 * \text{mAP}@.5 + w_2 * \text{mAP}@.75 + w_3 * \text{mAP}@.5 : .95$$

Extremely important to mention, *varying the Fitness Function does not mean to vary the parameters or hyperparameters of the network*; conceptually, given that the Fitness Function is used as the target of the *Early Stopping*, different Fitness Functions simply represent “different criteria” through which select the best set of weights. Experiments showed that YOLOv5s was very stable, i.e., by repeating the training different times without changing parameters, hyperparameters, and the Training/Validation Sets, they were obtained, at each epoch, about the same results (or results that were extremely close). This is also witnessed by the fact that some Fitness Functions achieved the highest value at the same epoch and they actually pointed to the same performances (see Table 6.13). Then, the results from all considered Fitness Functions were quite equivalent and, conceptually, all sets of weights could have been used indistinguishably. They were then adopted those related to the “original” Fitness Function (first row in Table 6.13). Additional details can be found in reference [117].

Network Testing and In-depth Experiments

As for the VGGish-inspired CNN, some tests were performed to:

- Analyse the contribution of TL and YOLOv5 augmentations.
- Evaluate generalisation performances on the Augmented Protection Barrier Test Set.
- Study the effectiveness of the approach in the extraction of the barrier’s motion.

Table 6.13. YOLOv5s Performances at different Fitness Functions

Fitness Function			Maximum Value	Best Epoch	mAP@.75	mAP@.5:.95	Total Loss
w_1	w_2	w_3					
0.1	0.0	0.9	0.98814	229	0.99499	0.98738	0.50670
0.1	0.2	0.7	0.98966				
0.1	0.4	0.5	0.99118				
0.0	0.3	0.7	0.98966				
0.0	0.5	0.5	0.99118	223	0.99500	0.98737	0.50599

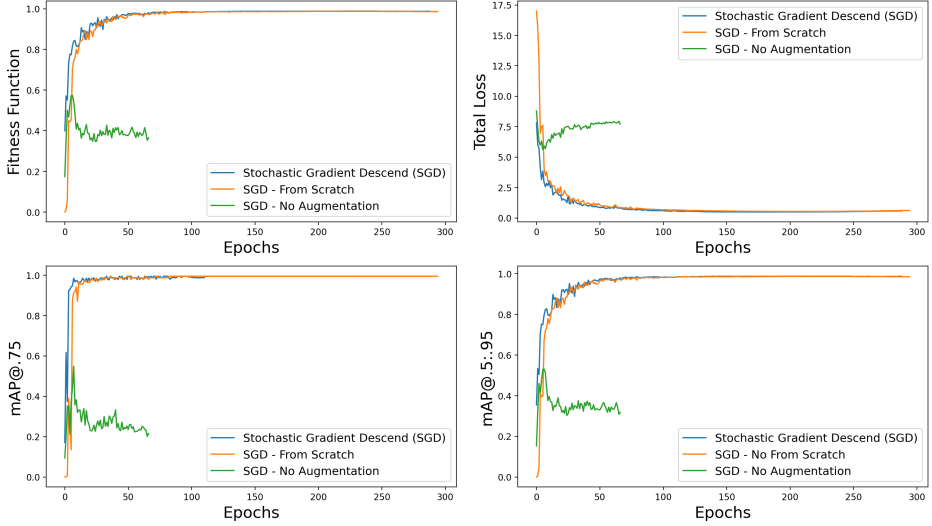


Figure 6.17. Contribution of [TL](#) and Augmentations in YOLOv5s

Advantages of Transfer Learning and YOLOv5s Augmentations.

Two experiments were carried out to test the advantages introduced by [TL](#) and by the augmentations that come with YOLOv5s. The first test expected to apply augmentation while performing training from scratch. Conversely, within the second test, augmentations were not considered but [TL](#) (by loading but not freezing the weights) was applied. Figure 6.17 shows the performances of YOLOv5s evaluated on the Augmented Protection Barrier Validation Set; both tests were performed by considering the original Fitness Function and the Stochastic Gradient Descent Optimizer. The blue lines in Figure 6.17 are exactly the same as those in Figure 6.16, they have been reported again to facilitate comparisons.

As can be noticed, YOLOv5s Augmentations drastically improve network performances, while [TL](#) does not contribute significantly. The latter aspect may be due to the fact that the addressed task (i.e., single object detection with an object that occupies a large portion of the image given the pre-processing in Section 6.3.2) is much easier than those for which YOLOv5 was developed (e.g., multi-object detection with objects at different scales); hence, the collected dataset seems to be large enough to let

YOLOv5s reaching adequate detection performances. Nonetheless, as can be seen from Table 6.14, by applying TL, they can be achieved slightly better performances.

Performances on Test Set. YOLOv5s was eventually tested on data it had never seen during the training phase. Table 6.15 shows the performance of YOLOv5s evaluated on all Augmented Protection Barrier Sets (see Section 6.3.2) when trained by considering the Stochastic Gradient Descent Optimizer and the original Fitness Function. Values for Training and Validation Sets are those related to the best epoch, i.e., the epoch when the *Fitness Function* achieved the maximum value.

On the Test Set, the network achieved quite high performances validating the initial hypothesis that YOLOv5s could have been re-used for this task even without performing in-depth parameters and hyperparameters tuning, differently from what happened with the VGGihs-inspired CNN (see Sections 6.4.1).

Effectiveness in Barrier Motion Extraction. To evaluate the effectiveness in barrier motion extraction, four video clips were taken into account which were extracted from the *DayClear* and *NightClear* GTA videos (see Section 6.3.2). These clips were directly pre-processed to extract, from each frame, only the patch related to the barrier (see Figure 6.7 for

Table 6.14. YOLOv5s Performances on Validation Set with/without TL

TL	Best Epoch	mAP@.75	mAP@.5:.95	Total Loss
Yes	229	0.99499	0.98738	0.50670
No	235	0.99496	0.98671	0.53968

Table 6.15. YOLOv5 Performances on Training, Validation, and Test Sets

Augmented Protection Barrier Set	Precision	Recall	mAP@.75	mAP@.5:.95
Training	1	0.984600	0.994990	0.991080
Validation	1	0.974760	0.994990	0.987380
Test	1	0.993372	0.994996	0.992064

clarifications). Having stated that, the considered video clips were:

- *DayClearOriginalClip*: a 28-second video clip (at 30 FPS) extracted from the DayClear video depicting the barrier during its motion (open-moving-closed-moving-open). Such a clip was composed of 842 patches¹⁴.
- *NightClearOriginalClip*: as the above, but extracted from the NightClear video. This clip contained 858 patches.
- *DayClearTransformedClip*: this basically contains the same patches of DayClearOriginalClip but they are transformed by randomly applying one of the transformations in Table 6.2 to each of them.
- *NightClearTransformedClip*: as the above, but starting from NightClearOriginalClip patches.

The first two clips i) did not contain any noise in terms of transformations and ii) contained some patches (specifically 240, see Section 6.3.2) which were already used in the Training, Validation, or Test sets. Therefore, these have been considered only to understand how the *Protective Barrier Motion Extraction Pipeline* (shown in Figure 6.2) would have performed in ideal conditions. Differently, for the last two clips – since the transformation for each patch was picked randomly for each patch and, also, augmentations themselves are based on parameters selected randomly from some specific intervals – it was assumed that it would have been extremely unlikely that the obtained transformed patches were identical to those considered within the Training, Validation, and Test sets. Hence, they were used to evaluate the actual detection performances of the *Protection Barrier Motion Extraction Pipeline*.

Figures 6.18 and 6.19 show the movement of the barrier obtained by processing the video clips through the pipeline; the first one compares the results obtained on *DayClear* clips, while the latter compares those related to the *NightClear* ones. Besides some small spikes, it can be seen that the movement of the barrier can be properly characterised through the

¹⁴Even though the images composing a video clip are generally referred to as “frames”, in this case, for the sake of consistency, it has been used the term “patches” to stress the fact that the original frames were pre-processed.

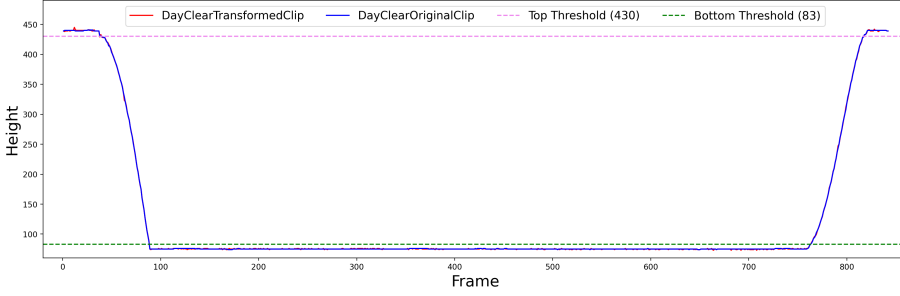


Figure 6.18. Barrier Movement Extracted from DayClear Video Clips

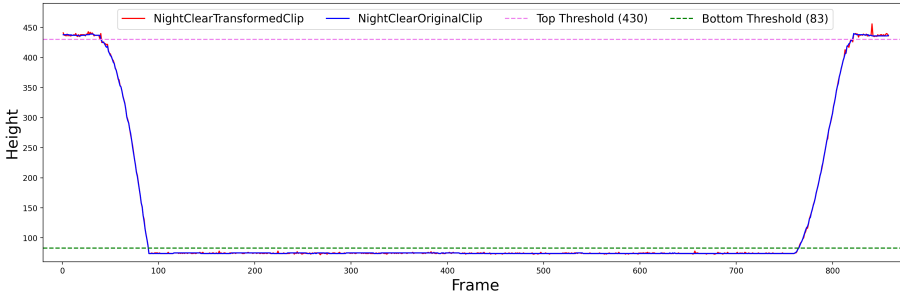


Figure 6.19. Barrier Movement Extracted from NightClear Video Clips

implemented pipeline. Important to mention, the height ('y' axis) is not related to the height of the barrier from the ground, instead, it is the height of the bounding box containing the barrier predicted by YOLOv5s (for each patch). Also, the two dashed horizontal lines represent some thresholds (empirically identified) which can be leveraged to identify the position of the barrier: i) if the height is above the *Top Threshold*, the barrier can be considered *open*; ii) if the height is below the *Bottom Threshold*, the barrier can be considered *closed*; iii) if the height is in between the two thresholds, the barrier can be considered *in movement* (or *moving*).

There are, however, some cases in which YOLOv5s is not able to detect the barrier (False negatives):

- In both *DayClear* clips, the barrier is not detected in patches 34, 35, 36, 37, and 819 (shown in Figure 6.20), which translates into a 0.6% miss-detection rate (5 patches over 842).



Figure 6.20. Undetected Barriers from DayClear Clips

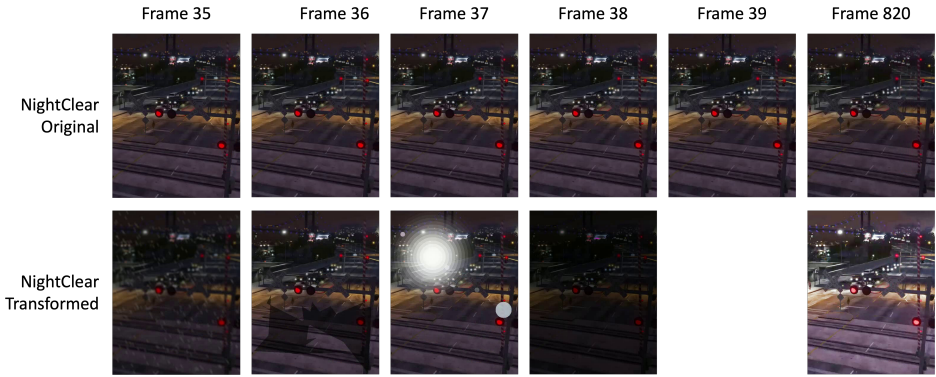


Figure 6.21. Undetected Barriers from NightClear Clips

- In *NightClearOriginalClip*, YOLOv5s was not capable of detecting the barrier in patches 35, 36, 37, 38, 39, and 820 (0.7% miss-detection rate, 6 patches over 858). About the same happened for *NightClearTransformedClip* but, in this case, frame 39 was correctly processed. These patches are shown in Figure 6.21.

Miss-detection occurred in both original and transformed video clips for almost the same patches, hence, it can be assumed that they are not due to augmentation. Instead, it can be due to the fact that the barrier, in those patches, was almost exactly vertical, while in other adjacent patches (given the camera angle) was slightly oblique.

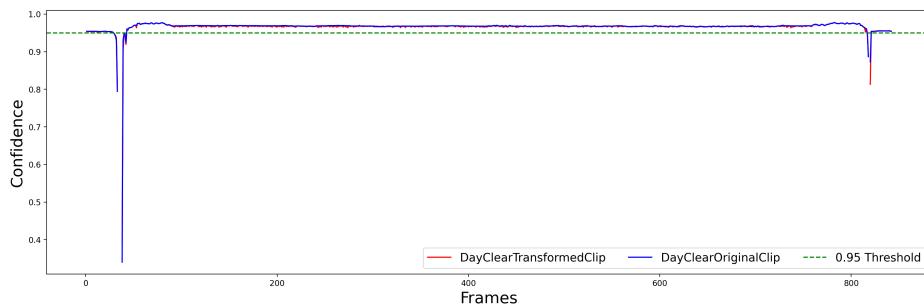


Figure 6.22. Prediction Confidences for DayClear Video Clips

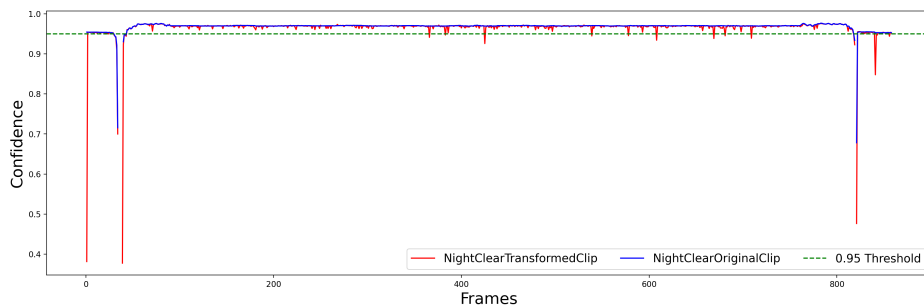


Figure 6.23. Prediction Confidences for NightClear Video Clips

This is also witnessed by the confidence that YOLOv5s associates with its prediction, reported in Figures 6.22 and 6.23 for each patch in the video clips. No confidence value was associated with non-detected barriers, while the 0.95 threshold has been plotted just to have a reference (high) value against which to evaluate predictions. Notably, besides some spikes, confidence values had similar trends:

- The confidence for each prediction is higher than 0.95 in most cases.
- When the barrier is open (first and last patches) the confidence is lower compared to when the barrier is moving or closed.
- When the barrier is visibly vertical, confidence values are much lower (low packs in the figures) compared to other cases, or, worse, the barrier is not detected.

To numerically quantify these differences, Table 6.16 reports the average confidence values for each video clip and for each position of the barrier (open, moving, and closed); to identify the positions, the Top and Bottom Thresholds as shown in Figures 6.18 and 6.19 were considered. Important to mention, as happened for *DayClearOriginalClip* and *DayClearTransformedClip*, patches intervals should have been the same also for *NightClearOriginalClip* and *NightClearTransformedClip*, as the position of the barrier is exactly the same in each patch of the pairs. However, it seems that the transformations disturbed the YOLOv5s detection, leading to predicted bounding boxes with slightly different heights; hence, the turning points (i.e., open→moving, and moving→closed) are slightly shifted in the case of *NightClearTransformedClip* compared to *NightClearOriginalClip*. However, these shifts were not significant (only a couple of patches); what is interesting, instead, are the average confidence values that seem to validate the hypothesis that YOLOv5s seems to struggle in correctly detecting the barrier when it is open, rather than closed or moving.

Then, for the sake of completeness, some specific augmentations could contribute further to lower prediction confidence. Figure 6.24 shows some of the patches related to the low packs of the red curve in Figure 6.23.

Besides the fact that it could be possible to try to improve network detection performances by adopting larger datasets (with more samples of any possible position the barrier can assume), the miss-detection problem can also be mitigated by implementing a corrective algorithm in post-processing. Assuming that YOLOv5s correctly detects the barrier in the i -th patch and its height is h_i , there could be three possible cases for the

Table 6.16. Average (Avg.) Confidence (Conf.) Values per Barrier Position

Barrier Position		DayClear OriginalClip	DayClear TransformedClip	NightClear OriginalClip	NightClear TransformedClip
Open	Patches	1:39 & 818:842	1:39 & 818:842	1:40 & 821:858	1:40 & 819:858
	Avg. Conf.	0.9373	0.9370	0.9458	0.9247
Moving	Patches	40:88 & 764:817	40:88 & 764:817	41:89 & 766:820	41:89 & 765:818
	Avg. Conf.	0.9714	0.9711	0.9711	0.9707
Closed	Patches	89:763	89:763	90:765	90:764
	Avg. Conf.	0.9681	0.9673	0.9702	0.9687
Overall Avg. Conf.		0.9606	0.9599	0.9614	0.9594

The annotation “X:Y” means “from patch X to patch Y”

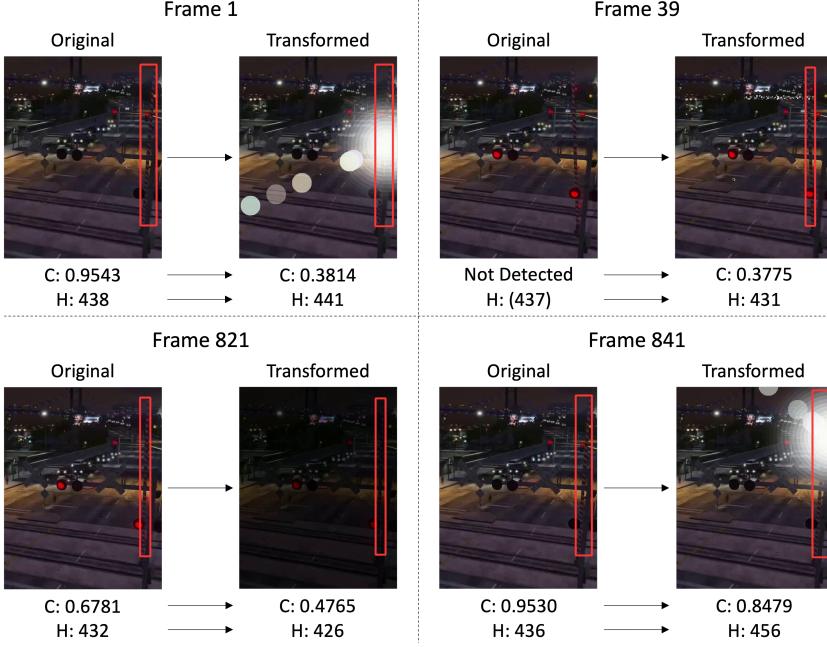


Figure 6.24. Patches from NighClear Clips with Low Confidence Values.
C: Confidence Value - H: Detected Height.

(i+1)-th patch:

1. *YOLOv5s predicts a single bounding box.* In this case, the predicted bounding box can either be a True positive (i.e., it contains the barrier) or a False negative (i.e., it does not contain the barrier). Since in adjacent patches the position of the barrier does not vary that much (especially if the camera has high FPS and it suddenly breaks), then, it can be established an interval to check the detection. For example, if the height of the bounding box extracted from patch (i+1)-th is within the interval $[h_i - K; h_i + K]$ (where K is a value that can be pre-specified empirically), then, the prediction can be considered correct; otherwise, it is not correct and should be discarded.
2. *YOLOv5s predicts more bounding boxes.* The same reasoning above can be applied to this case. It will be considered as the only True

positive the bounding box whose height is closer to h_i and falls within the interval defined above (if any). All others would be False positives and, thus, discarded.

3. *YOLOv5s does not predicts any bounding box.* In this case, a False negative occurs because in each patch there is a barrier, hence, at least a bounding box should be predicted. To mitigate this problem, it can be simply assigned to h_{i+1} , i.e., the height of the bounding box extracted from the $(i+1)$ -th patch, the same value of h_i . The same solution can be applied also in the first and second cases above, if the detection had to be *discarded*.

It is important to mention that, in the tests performed with the various video clips, only the third case occurred. Also, given the low miss-detection rates (0.6 / 0.7%), this solution was found to be viable even if YOLOv5s failed to detect bounding boxes in some adjacent patches (maximum 5). It was then applied to adjust miss-detection; this is why, in Figures 6.18 and 6.19, lines are not interrupted as happens in Figures 6.22 and 6.23. On the other hand, this issue should be deeply evaluated case-by-case in real environments as too many consecutive miss-detection could affect the extraction of the barrier movement and then lead to wrong analyses.

To conclude, it was also evaluated the *computation* time of the *Protection Barrier Motion Extraction Pipeline*. Table 6.17 shows the average computation times required by the pipeline to fully process a single patch; the miss-detection post-processing is included in “Height Extraction”. Given

Table 6.17. Pipeline’s Computation Times in milliseconds (ms)

Vieo Clip	Avg. Time per Patch			Total
	<i>Cropping</i>	<i>YOLOv5s</i>	<i>Height Extraction</i>	
<i>DayClearOriginal</i>	3.4493	6.9784	0.0018	10.4295
<i>DayClearTransformed</i>	3.4493*	6.8656	0.0021	10.3170
<i>NightClearOriginal</i>	3.3114	7.0391	0.0020	10.3525
<i>NightClearTransformed</i>	3.3114*	7.0193	0.0021	10.3328
Avg.	3.3804	6.9756	0.0020	10.3580

* patches were first extracted and then augmented, so cropping times are the same.

the low computation times, this approach could be suitable for real-time applications, at least for the extraction of the movement of the barrier.

6.5 Discussion of Results

Within this chapter, two DNNs have been analysed that could be leveraged to implement the Warning Bell and Protective Barrier Analysis modules in the context of the “LC Intelligent Monitoring System” proposed in Section 6.2.1. Here, all the obtained results are discussed together with additional proposals for the implementation of the modules that have not been addressed. Then, RQs and KPIs defined in Section 6.1 are respectively answered and evaluated.

6.5.1 Analysing Results and Future Implementations

Based on the performed tests and the argumentation presented throughout this chapter, all modules expected by the LC Intelligent Monitoring System in Figure 6.1 are discussed herein with the aim of evidencing possible criticalities to be overcome and/or opportunities for future improvements.

Warning Bell Analysis Module

A three-class classification approach (Section 6.2.2) was proposed for this module. It was implemented by: i) leveraging the AudioSet dataset and the YouTube platform to collect data (Section 6.3.1); and ii) building a VGGish-inspired [64] CNN to recognise *warning bell* sounds (Section 6.4.1). The CNN was trained on 1-second frames extracted from collected audios achieving an accuracy of 92.48% at the Frame-Level and of 97.37% at the accuracy at the Audio-Level (Section 6.4.1).

Given these results, it is deducible that the implemented CNN is not free from miss-classification errors. On the one hand, *False negatives*, i.e., not-detected warning bells, could lead to unnecessary interventions; on the other hand, *False positives*, i.e., *generic alarm* or *no alarm* samples classified as *warning bell*, may lead to missed maintenance activities. These issues are clearly of primary importance and should be taken into account when developing such a system for on-field applications, however, there are

some scenarios, which are not far from what would happen in reality, that could help to mitigate these problems:

- First, before activating repair interventions, the system's results could be manually checked by human operators who would either activate maintenance procedures or discard potential *False negatives*.
- Second, to analyse how *False positives* could be mitigated, two cases should be considered starting from the reasonable hypothesis that the *Orchestrator* could understand whether an LC is active or not by integrating information (e.g., the trigger event that activates the LC when a train is approaching) coming from external systems:
 - **The LC is not active.** It may happen that a warning bell is detected even if the LC is not active. In case it is a real malfunction, it would probably repeat multiple times; hence, after K bells detection while the LC is not active, it could be evaluated by human operators whether corrective interventions would be necessary. Otherwise, if it happens once in a while, it can be considered as a *False positive* and, somehow, ignored.
 - **The LC is active.** In this case, the problem is represented by the condition that the bell is actually defective, but it is still detected by the network. Also in this case, this condition may be mitigated by the aleatory nature of the environment itself and by the multiple activations of the LC during the day/week. It is unlikely that, each time the LC is active, there would be a sound misleading the classification performed by the CNN. Then, it would be reasonable to assume that, if at the i -th activation of the LC a *False positive* occurs, it will not occur again at the $(i+1)$ -th or $(i+2)$ -th activation.

Although the suggestion of improving network performances still remains to try to eliminate the issues above, compared with the current situation where LC malfunctions can remain undetected for a long time (given scheduled inspections), the proposed approach could represent an improvement in terms of both maintainability and safety, also considering a certain – reasonably limited – amount of *False negatives and positives*. Actually, important underlining, if the *warning bell* class is considered as the

positive one, while the *no alarm* and *generic alarm* classes are considered as the *joint* negative class, the CNN did not produce any *False negative* or *False positive* at the Audio-Level (see Table 6.7). Miss-classification errors only happened between *no alarm* and *generic alarm* classes.

In addition to the analyses above, there are other considerations related to the network classification performances that are worth mentioning:

- **Transfer Learning and Dataset Dimension.** TL offers good trade-off between network performances and network stability. Additionally, it was also identified the correlation between the dimension of the training set and network classification accuracy: as a general rule, the larger the training set the better the performance. These results seem to indicate that if, on the one hand, TL helps to improve performances when only limited data are available, on the other hand, there may be still room for further improvements if additional training samples are collected.
- **Cross-country Performances.** As expected, the network was not capable of correctly classifying warning bells from a specific source (country, in the analyses above) if data from the same source were not included in the training set. Although the performed tests may not be comprehensive enough to generalise findings, results seem to suggest that, in order to achieve high accuracy, it would be advisable to train the CNN with samples related to the warning bell it should detect at run-time; in the same way, *generic alarm* samples should particularly include sounds that could be confused with the warning bells under examination.
- **Robustness to Noise.** As a general rule, tests showed that the higher the contribution of the noise, the lower the classification accuracy.

By focusing on maintenance technical aspects, to try overcoming noise-related issues, it would be possible to adopt, among others, two main solutions. The first one would be to augment training data by adding noisy samples so that the CNN would be potentially able to correctly classify them. The second one, instead, would expect to build a supportive model oriented at detecting noises. Its outcome could

then be used to weight CNN predictions, hence, if there is too much noise, the CNN prediction could be considered as “partially reliable”. Nonetheless, the same hypothesis on environment aleatory drafted above could hold also in this case: it would be quite unlikely that, at each activation of the LC, there would be consistent noises affecting audio recordings; there would then be cases, potentially occurring with higher frequency than current on-field scheduled inspections, where the CNN would operate correctly given the absence of noise.

To conclude, there is another interesting aspect to consider which is related to safety rather than maintenance. One of the main purposes of warning bells is to notify road users and pedestrians of the incoming train at LCs. *If the CNN miss-classifies the warning bell given intense noise, it may be a symptom of poor signal audibility also by road users.* This would clearly imply that the CNN is capable of approximating, to some extent, human hearing; but this aspect has not been addressed so far. However, it could be taken into account as future research direction by considering the following:

“Although LC components (specifically warning bells and lights) operate properly, it is not said that they would be correctly visible or audible from the perspective of road users given adverse environmental conditions (e.g., harsh weather phenomena or noisy events). These, in conjunction with other possible malfunctions, may lead to unpleasant consequences. Therefore, it would be possible to study the correlation between DL models predictions and human perception to evaluate if additional protective measures would be required to avoid incidents at LCs in case of unfavourable environmental events.”

Protective Barrier Analysis Module and Orchestrator

To implement this module, YOLOv5s (Section 6.2.2) was adopted. Data to train the DNN were collected through GTA and pre-processed by exploiting the task-specific characteristic of *fixed camera*; then, data augmentation was applied to increase dataset dimensions and introduce aleatory (Section 6.3.2). Eventually, the trained YOLOv5s (Section 6.4.2) achieved an overall mAP@.75 of 0.99499 and an average confidence of 96.05% on test data. Nevertheless, also YOLOv5s is not free from producing *False*

negatives (and, probably, also *False positives*) under specific conditions. However, also in this case, they do not represent significant threats as they can be easily mitigated by applying some corrective algorithms as widely analysed in Section 6.4.2; these, together with a low miss-detection rate (0.6-0.7%), make the proposed network viable for barrier detection.

A clarification is necessary at this point. YOLOv5s achieved extremely promising performance; the motivations behind the choice of this DNN, and the purpose of these analyses, have been given at the beginning of this chapter. However, it is required to emphasise the fact that there may be other, less complex DL models, or even traditional CV techniques, that might achieve comparable performance while requiring less computational power. An example of less complex DNN may be YOLOv5n (the ‘smallest’ YOLOv5 version), which could be evaluated and compared against the results obtained in this Thesis; alternatively, approaches like knowledge distillation [53, 16] could be exploited to conceptually “infuse” the knowledge acquired by a larger model (YOLOv5s in this case) into a less computational-intensive DNN. Lastly, also approaches based on traditional CV and/or ML algorithms (e.g., [41, 91]) could be investigated; even thought, from the literature (see Sections 2.1.3 and 2.3), DL approaches seem to have widely surpassed traditional CV ones. In-depth optimisations and comparative analyses of this kind have not been the subject of the studies discussed in this Thesis, but they would be fundamental aspects to take into account when developing AI systems for operational purposes, as they would allow for the identification of the optimal trade-off between performance and required computational power.

Having stated that, after the movement of the barrier is extracted, this information should be processed to understand if the barrier is functioning correctly. This analysis would be performed by the *Orchestrator* which, as anticipated, besides gathering the output of the other modules of the “LC Intelligent Monitoring System”, could also integrate information coming from external systems (e.g., the moment when the LC activation is triggered given the incoming train). This would make possible the detection of the eventual delay between the trigger event and the activation of the barrier detected by the Protective Barrier Analysis module. Hence, some mechanisms can be adopted to estimate the degree of malfunction of the barrier.

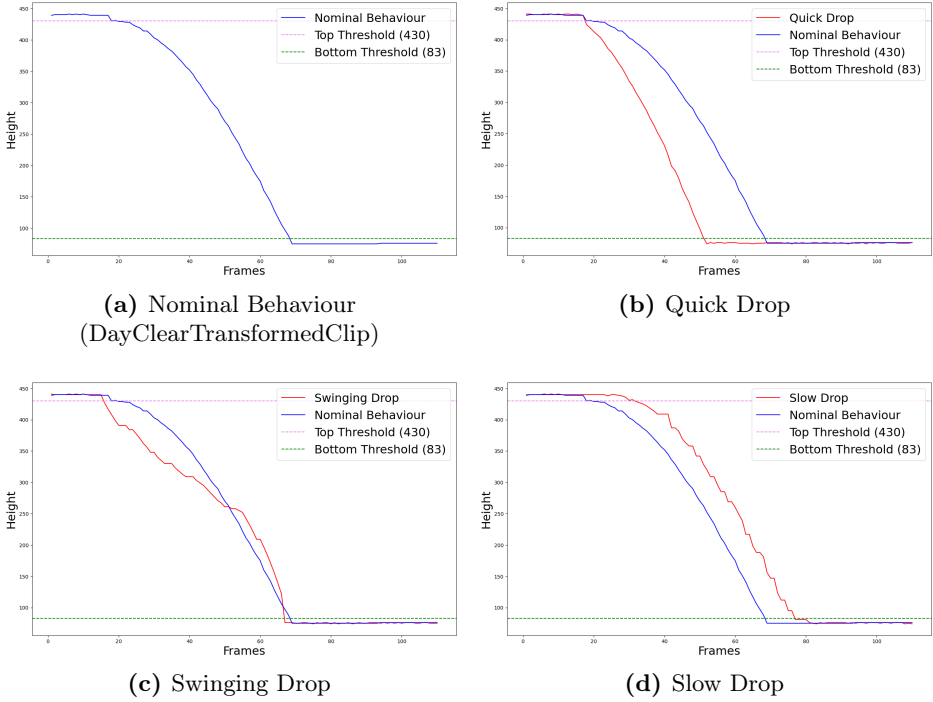


Figure 6.25. Simulated Barrier's Malfunctions

The implementation of the *Orchestrator* has not been addressed in this Thesis, however, they are discussed herein some possible solutions that could be adopted to understand whether the barrier is defective or not; similar solutions could be adopted for warning lights analysis.

As the first step, some malfunctions were simulated by randomly modifying the nominal behaviour of the barrier. For these tests, it was considered the nominal behaviour of the barrier extracted from the *DayClearTransformedClip* (i.e., the red line in Figure 6.18); also, it was taken into account only the first phase of the movement, i.e., the transition from open to closed as shown in Figure 6.25a. Then, the following malfunctions were simulated: i) *Quick Drop* (Figure 6.25b), which simulates the case when the barrier closes too quickly or starts closing earlier compared to the nominal behaviour; ii) *Swinging Drop* (Figure 6.25c), that represents the case when

Table 6.18. Metrics for Anomaly Evaluation in Barrier Movement

Barrier Movement	Start Frame	End Frame	Time	Start Delay	End Delay	Time Taken (%)	MSE
<i>Nominal Behaviour</i>	17	67	50	/	/	/	/
<i>Quick Drop</i>	17	51	34	No	- 16	- 32.0 %	4623.0
<i>Swinging Drop</i>	15	66	51	- 2	- 1	+ 1.9 %	648.9
<i>Slow Drop</i>	31	76	59	+ 14	+ 9	+ 15.2 %	2272.9

MSE: Mean Squared Error

the barrier does not close smoothly as it should but takes about the same time as in the nominal case; and iii) *Slow Drop* (Figure 6.25d), which simulates the case when the barrier closes too slowly or it starts closing too late. In these examples, it was taken into account the instant when the curve representing the nominal behaviour crossed the Top Threshold as the reference point identifying the event “the barrier is closing”; similarly, the Bottom Threshold was considered to identify the event “the barrier has closed” (see Figure 6.18 and related explanation). In a real scenario, when possible, it could be exploited the trigger event produced by the approaching train as the reference point against which to perform the following evaluations. On the other hand, for the characterisation of the detected behaviour, those above could represent suitable assumptions to detect the starting and ending points of the different phases of the barrier movement.

Then, by considering the events above, it would be possible to obtain information like those reported in Table 6.18. The values under the “Time” column are expressed in frames; hence, considering that videos have been collected at 30 FPS, it can be deduced that, in the nominal case, the barrier closes in ~ 1.66 (50/30) seconds. Lastly, metrics like those in the Table, e.g., “Time Taken (%)” and the “Mean Squared Error” can be leveraged to define the health status of the barrier.

Warning Light Analysis Module

As from Section 6.2.2, the Warning Light Analysis module was not addressed in this Thesis as classification approaches, identified to be suitable solutions for warning light behaviour extraction, have already received

much attention in the literature. Indeed, as it happened with the barrier, it would be possible to extract the behaviour of warning lights over time by evaluating, frame by frame, their status through a CNN which would “simply” classify them in “turned on” or “idle”

Then, as mentioned in the previous section, similar solutions to those presented for protective barrier health status analysis can be adopted also in this case: once the behaviour at run-time is extracted, the *Orchestrator* could implement mechanisms to understand whether warning lights are behaving by following the nominal pattern or not.

6.5.2 Answering RQs and Evaluating KPIs

Given the above experimentation, this section tries to formally answer to the RQs and evaluate the KPIs set in Section 6.1.

Research Questions

RQ1: *How would the above characteristics affect or support the implementation of a continuous monitoring system based on DL?*

“Above characteristics” refers to the fact that: i) LCs themselves could be partitioned into subsystems: ii) the monitoring camera has a fixed field of view; iii) warning bells from different sources may have different sounds.

As for the first one, it directly allows for the exploitation of a modular approach. Multiple modules, one for each subsystem, could be implemented as widely discussed in this chapter; this would then allow for the separation of concerns between different modules. Hence, if one of the modules is faulty, only the analysis of a single subsystem would be (hopefully temporarily) compromised.

The second characteristic also helps in this direction. By exploiting the fixed cameras, it has been possible to apply a pre-processing step that allowed for the facilitation of the tasks the vision-based modules had to address.

Lastly, the third characteristic could actually be seen as a threat to the system. Tests have shown that it would be better to adopt a specific classifier for each warning bell source.

RQ2: How could the problem of data availability be addressed? How can suitable datasets be generated (even synthetically) for the development of a PoC in this context?

Data availability is a quite sensitive issue, as DNN often require a lot of data to be properly trained and to perform efficiently. This has also been demonstrated with the VGGish-inspired CNN discussed in Section 6.4.1 where *TL* played a central role in achieving good performances on a relatively small dataset. *TL* would be a means to cope with the limited data problem, but “target data” would still require to be collected somehow.

For the development of this PoC, AudioSet, YouTube, and GTA resulted to be relevant for easy data collection. Nonetheless, these strategies would be advisable only for PoC development. Once a suitable strategy has been set, and the DNNs identified and proven to be suitable, tests on real data from the real operational environment where the system should work would be required.

RQ3: Is it actually possible to monitor and detect anomalies in LC sub-systems without using intrusive sensors? Specifically, is it possible by leveraging only cameras and microphones?

The DNNs presented in this chapter resulted to have quite high performances in detecting LC components (i.e., warning bell and barriers). Also, especially in the case of barriers, it has been demonstrated that their movement can be extracted without adopting intrusive sensors. Therefore, it would not be so visionary to think that data extracted from such sensors and processed through DL approaches would be extremely useful in defining the health status of LCs.

Key Performance Indicators

KPI1: Implementability Costs. *Includes sensors and installation costs.*

Based on the vision set in this chapter, cameras and microphones would allow for the monitoring of LC at quite contained costs. First, being non-intrusive, they could be installed externally without running into eventual re-approval processes. Second, their re-usability, i.e., exploitable for both maintenance and safety application [42],

makes them even more cost-effective as, with a “single” installation, multiple crucial aspects can be monitored. Third, there may be cameras (maybe smart cameras with integrated microphones) already installed at LCs for security purposes; those could be exploited also to implement what is proposed in this chapter, hence, no installation costs would be required.

KPI2: Computation Time. *Refers to the time required by the system to detect possible malfunctions.*

The VGGish-inspired CNN is capable of analysing a 1-second warning bell frame in 1.13 ms, while the pipeline to extract the height of the barrier from each frame takes 10.36 ms per frame (clearly, on average). Whether YOLOv5s’ performances are adequate or not might depend on the FPS of the camera, e.g.: if the camera works at 30 FPS, it will capture a frame each 33.3 ms, while it will capture a frame each 16.6 ms in case of 60 FPS. Then, in case it would be implemented real-time monitoring, to efficiently exploit the sensor, the system should be capable of processing frames at a higher FPS rate than the camera. In this case, 10.36 ms is lower than both the thresholds indicated above, hence, *for what concerns barrier detection*, the performances are in line with real-time requirements at 30 or 60 FPS. Also, some margin remains for the implementation of the anomaly detection algorithm (or DNN) that would evaluate the health status of the barrier or estimate its remaining useful life.

KPI3: Effectiveness. *Indicates performances of the approach in terms of accuracy and detectable malfunctions.*

The implemented DNN are capable of detecting *warning bells* or *protective barriers* with quite high “accuracy”. To understand whether it would be possible to efficiently detect malfunctions through the proposed system, however, further developments would be required.

6.6 Structured Evaluation of the Approach

As per Chapter 5, this section summarises the relevant aspects discussed through this chapter in relation to the proposed approach. These, together

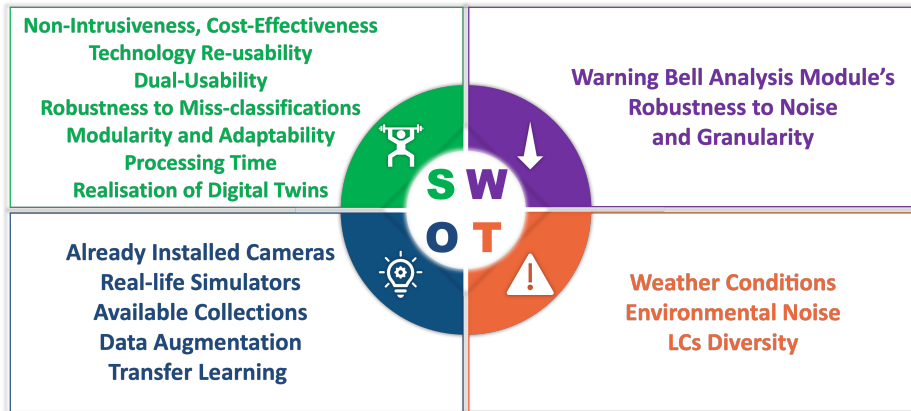


Figure 6.26. A SWOT Analysis for the *LC Intelligent Monitoring System*

with other interesting facets, are arranged by following the structure of the **SWOT** analysis (see Figure 6.26) so that they can be easily evaluated.

6.6.1 Strengths (↑) and Weaknesses (↓)

Non-Intrusiveness, Cost-Effectiveness, and Technology Re-usability

(↑). The *LC Monitoring System* discussed in this chapter is thought to work with *cost-effective* and *non-intrusive* sensors, specifically, microphones and cameras. The *cost-effectiveness*, as mentioned in Section 6.5.2, derives from three factors: i) *non-intrusiveness*, since it is possible to install these sensors without modifying the asset in any manner, no re-approval process would be required; ii) *re-usability of the sensors*, as these sensors can be used both for maintenance and safety (e.g., obstacle detection) purposes, hence, installation costs would be addressed only once; iii) *technology re-usability*, as (smart) cameras already installed at *LCs* can be exploited also for maintenance/safety purposes.

Dual-usability (↑). A system of this kind, exploiting cameras and microphones, could bring benefits in terms of both *maintenance* and *safety*. As mentioned in Section 6.5.1 in the context of the Warning Bell Analysis module, if properly developed, *DL* models could somehow

approximate human senses. Hence, besides monitoring the health status of LCs, the system can also be adopted to understand the perceptibility of warning signals. For example, the VGGish-inspired CNN implemented for warning bell classification resulted to be sensitive to intense noises (see Section 6.4.1), producing wrong classifications. This, despite being a weakness from the perspective of maintenance, could actually open opportunities for safety concerns as the misclassification of a warning bell could be a symptom of poor audibility of the signal. This suggests the potential *dual-use* of the proposed approach which, as mentioned in Section 6.5.1, should be clearly further investigated in the future to be validated.

Robustness to miss-classifications (↑). As analysed in Section 6.5.1, for the implemented approaches (i.e., warning bell and protective barrier detection), there could be applied some corrective mechanisms or expedients to mitigate or overcome possible miss-classifications.

Modularity and Adaptability (↑). Being a modular approach, it could be easily re-adapted to any kind of LCs, whether they have all subsystems (i.e., warning bells and lights, and protective barriers) or only a subset of them. In addition to that, another advantage of this modular approach is given by the fact that, by exploiting cameras and microphones, it focuses on the perceptible behaviour of the LC, considering it as a *black box*. In other terms, instead of focusing on LC components' specifics, the proposed monitoring system focuses on their audible and visible behaviour. Then, it should not be intensely re-calibrated each time it has to be used for a different type of LC. The DL models, their architectures, the approaches, and the reasoning would remain unchanged; what could be done is a *fine tuning* of the DNNs on data related to the new type of LC.

Processing time (↑). Implemented networks have quite low processing time potentially allowing, depending on future developments, for the real-time analysis of the LC status.

Realisation of Digital Twins (↑). This is an aspect that has not been deeply analysed in this chapter, however, deserves to be mentioned. As recalled in Section 2.1.1, a DTs can be seen as a digital representation

of a physical asset which is connected to it by means of (IoT) sensors and evolves with it over time [23]; in this context, AI/DL could bring several benefits. Focusing on the proposed monitoring system, it can actually be exploited to (partially) build LC DTs. For example, instead of using intrusive IoT sensors to retrieve the movement of the barrier, the combination of video data and DL (in this case, the whole Protective Barrier Analysis module) can be used as a *macro/advanced intelligent sensor* to extract, in real-time, data that the DT would need to model the behaviour of the barrier.

Warning Bell Analysis Module’s Robustness to Noise and Granularity (↓). The robustness to noise has been widely analysed in Section 6.4.1; as the general rule, the higher the noise intensity the worse the classification performances. Despite being potentially useful from the perspective of safety to evaluate the audibility of warning bells, and despite being possibly mitigated by the aleatory of the environment (it is unlikely that at each LC activation there would be noises disturbing audio detection), it is a matter of fact that noises could affect the performances of the implemented VGGish-inspired CNN. Hence, further improvements should be made to improve the robustness of the Warning Bell Analysis module to environmental noises.

Another important aspect is related to the *granularity* of the module. The CNN works with 1-second frames, however, it would be visionary possible to improve this granularity by, e.g., trying to make the DNN capable of analysing frames with smaller duration (perhaps in the order of milliseconds). Hence, it could be implemented an approach similar to that proposed for warning lights or barriers, i.e., extract the behaviour of the warning bell by plotting predictions over time. This is already possible, but with a prediction each second; with finer granularity, the behaviour could be modelled more properly.

6.6.2 Opportunities (↑) and Threats (↓)

Already Installed Cameras (↑). As already mentioned, cameras may have been already installed for security (most commonly) or safety (e.g., obstacle detection [42]). This would make this approach im-

plementable without installing additional cameras (and, perhaps, microphones).

Real-Life Simulators and Available Collections (↑). One of the main challenges to be addressed concerned *data availability*. For PoCs development, real-life simulators (like [GTA](#)), data collections available online (like AudioSet), or multimedia platforms (like YouTube), offer the possibility of collecting real-like synthetic and/or real-world data in a timely manner.

However, it is still important to underline that the datasets obtained through the collection methods above may not be fully representative of reality related to a specific task. They would be extremely useful for PoCs development, experiment DNNs models, and define suitable strategies to address a given problem; however, to evaluate the real effectiveness of the implemented approach, real data should be used, possibly collected directly from the assets to be monitored by following ad-hoc procedures and considering specific requirements (e.g., the angle of the camera or the distance of the microphone from the [LC](#)).

Data Augmentation and Transfer Learning (↑). In addition to methods for timely data collection, some techniques can be used to increment the amount of collected data (*data augmentation*) or to achieve good performances also on a limited amount of data (*Transfer Learning*).

Weather Conditions and Environmental Noise (↓). Weather phenomena and environmental noises may affect the performances of the proposed [LC Intelligent Monitoring System](#) as particularly happened with the Warning Bell Analysis module (see Section [6.4.1](#)). On the other hand, the Protective Barrier Analysis module resulted to be quite robust to weather phenomena or other disturbing events (sun flares, holes, etc.).

Besides the specific [DNN](#), there is another aspect to take into account. The assumption made at the beginning of this chapter was to exploit the *fixed camera* characteristic. They would be quite rare but there might be harsh phenomena that could shift the field of view of the

camera. However, although rare, they are worth mentioning as, in that case, some physical re-calibration would be required.

LCs diversity (↓). As mentioned above, the approach is easily adaptable to almost any type of **LCs** as it does not consider specific **LC** implementations but focuses on their visible and audible behaviour. Nonetheless, it is important to point out that warning bells may have quite different sounds according to the type of the **LC** to be monitored. This could potentially become a threat in case there would be particular warning bell sounds that would be extremely challenging to be detected against similar acoustic events. Hence, particular attention should be posed while adapting the Warning Bell Detection module.

6.7 Identification of Future Directions and Recommendations

To conclude with the analysis of this case study, some general directions and recommendations are reported herein, that would potentially benefit future research in this context.

Combine Audio-Video Data with **DL to Generate Digital Twins.**

A **DT** requires to be connected to its physical counterpart by means of (**IoT**) sensors giving it the possibility of evolving in real-time with the physical assets. Nonetheless, when it comes to assets monitoring, **IoT** sensors are typically *intrusively* installed on the asset to be digitalised. Some of the assets recently deployed may already be equipped with the latest sensors and, hence, are predisposed by design. On the other hand, assets deployed decades ago would require massive sensorization to be digitalised. In the case of **LCs**, as already explained, it would be possible to leverage the proposed monitoring system to also generate **LC DTs**. Then, instead of using intrusive sensors to monitor physical parameters (e.g., current signals), it would be possible to monitor the visible and audible behaviour of the components. Clearly, the parameters associated with the engine that moves the barrier and the movement extracted from video data are not exactly the same

information, however, malfunctions in the engine would most likely reflect in a non-smoothed behaviour of the barrier. Therefore, both of them can be exploited for the same purpose, e.g., to understand whether the barrier is defective or not. Hence, audio and video data can be pre-processed through DL and passed to the DT [23] so that the virtual components would behave as the physical ones.

Real-Life Simulators for Synthetic Data Generation. Despite real data being always the best choice, there are cases in which their collection is not possible or, it is not possible to timely collect all possible events. In these cases, for PoC development, software like video games could be exploited to timely collect data which would also sufficiently represent reality.

Facilitate AI Tasks. Instead of adopting a single DL model to solve a complex task, it is advisable to subdivide it, if possible, into multiple, easier sub-tasks. Each of them can then be handled by a separate DL model, specifically designed for its resolution. In this direction, it would be advisable to exploit problem-specific characteristics, as explained in Section 6.2.1. In that case, it has been possible to exploit the *fixed camera* to split the problem of warning lights and barrier detection into two tasks, both of which could have been further simplified: i) the warning lights behaviour extraction was turned into a classification problem, instead of an object detection one; ii) the object detection oriented at detecting the barrier had to look for it in a smaller portion of the image.

Exploit DL to Increase Safety at LCs. Although the case study discussed in this chapter focused on LC maintenance, it would not be visionary to assume that, if properly developed and integrated within a fully-connected environment (see, for example, GoI4 in Chapter 3), a solution of this kind could be adopted also to improve safety at LC. A possible scenario, as already anticipated in Chapter 3, would be: a train can be notified with the health status of the LC it is approaching so that it could autonomously adopt proper countermeasures in case the LC is predicted to be defective.

Another way to possibly improve safety would be to adapt what discussed in Chapter 5 about obstacle detection, or other existing

solutions (e.g., [42]), to check whether the LC area is free from obstacles. Actually, in this case, the same camera sensors might be used for both maintenance and safety purposes.

To conclude, DL approaches analysing audio and video data could also approximate human behaviour. For example, as discussed in Section 6.5.1, misclassifications of the Warning Bell Analysis module given the noisy environment could be a symptom of the fact that the bells are not properly audible for road users as well. This aspect might be quite relevant and could be exploited to analyse warning bells from the perspective of safety if further investigated.

Conclusions

In this Thesis, the problem of integration of **DL** in railways was analysed, evaluating opportunities and challenges. First, a definition and a reference taxonomy for **AI** in railways were proposed (Chapter 2), with the aim of providing railway researchers and practitioners, having no or limited **AI** background, with a set of basic concepts facilitating their approach to the field. Then, challenges and opportunities were evaluated from both methodological and experimental perspectives.

First, in order to support the step-wise integration of **AI** in autonomous train driving systems, some **Grades of Intelligence (GoIs)** were proposed (Figure 3.2) which, building upon well-established *Grades of Automation (GoAs)*, would help to define some high-level guidelines allowing for the *conceptual shift* from *automatic* to *autonomous* trains in *open* environments (Chapter 3). In this process, certifiability challenges were considered, with the *Safety Envelope* (Chapter 2) being identified as one of the key paradigms that could enable the integration of **AI** in train operations in the coming years. Then, *structural needs* were identified and an example architecture was drafted (Figure 3.3), structuring the vision for future full Autonomous Trains in fully connected railway environments.

Then, some guidelines to support the development of **DL** modular architecture for railway safety and maintenance applications were systematised (Chapter 4). Such a methodology details seven steps from the definition of the problem statement (with the identification of the domain- and problem-specific peculiarities being one of the crucial factors) to the structured

evaluation of the approach and the delineation of future research directions. Hence, two case studies were addressed, which were used as stages to exemplify the adoption of the systematised methodology and propose solutions to cope with the limited availability of data.

The first case study (Chapter 5) concerned the identification of a suitable approach for the detection of any possible obstacles (whether known or unknown in advance) on rail tracks while exploiting the minimum onboard equipment, i.e., one/multiple cameras (given the scalability of the approach). A multi modular architecture was then proposed (see Figure 5.1), with the *Rail Tracks Identification* and *Anomaly Detection* modules being implemented and discussed. Tests showed that unsupervised anomaly detection approaches would potentially be viable to extend the coverage (intended as type and number of localisable obstacles) of vision-based obstacle detection systems, typically addressed in the literature through object detection approaches and, thus, capable of identifying only pre-defined types of obstacles. Also, the domain-specific characteristics concerning the *fixed and pre-determined train paths* seemed not to be fully exploited in the literature. Therefore, experiments were performed to evaluate the opportunities it could introduce. Despite tests may not have been comprehensive enough to generalise findings, it was showed that its exploitation would be potentially crucial in properly identifying the correct rail track ahead the train in case of multiple rail tracks within the captured scene.

The second case study (Chapter 6) concerned the definition of a strategy for the continuous monitoring of **LCs** without interfering with their operations. A modular architecture was proposed also in this case (see Figure 6.1), where each module focuses on the monitoring of a specific **LC** component; specifically, the Warning Bell and Protective Barrier Analysis modules were implemented. The exploitation of problem-specific characteristics was relevant also in this case, allowing for the simplification of the tasks that the **DNN** had to address. Tests showed that the motion of the barrier is correctly extracted, visionary paving the way for future anomaly detection approaches oriented at evaluating the health status of the component. Other interesting results come from the tests conducted to evaluate the robustness to environmental noises of the **CNN** implemented for warning bell detection. In general, they suggest that the higher the

contribution of noise the worse the classification performances of the CNN. Although this could be seen as a shortcoming in terms of warning bell detection and, thus, maintenance activities, this finding could visionary open huge opportunities in terms of safety: misclassification due to noise in the environment could be a symptom of poor audibility of the warning signal also by road users. Therefore, in case this aspect would be further investigated and validated, a solution of this kind could potentially have dual applicability: on the one hand, monitoring the health status of the LC, on the other hand, evaluating the perceptibility of warning signals.

Both case studies played a key role in understanding how the problem of data availability could have been overcome. In this context, techniques like Transfer Learning or data augmentation resulted to be extremely useful to obtain suitable results even on a limited amount of collected/available data. At the same time, in case no data were available, 3D Editors (e.g., RoadRunner and Unreal Engine) and Real-Life Simulators (e.g., GTA V) were found to be optimal allies in the timely and safe collection of proper datasets for the development of DL Proofs-of-Concept, offering a good approximation of the reality in terms of both graphics and physics of the agents. Two virtual railway scenarios were built, to collect proper data to train and test the implemented DNN in the context of the Obstacle Detection case study. Hence, to partially contribute to the data availability problem, the datasets generated through RoadRunner and indications for the construction of a dataset for warning bell classification were published.

In conclusion, the Thesis analysed the opportunities that DL, with a specific focus on audio-video analyses, could bring in the contexts of safety for autonomous trains and maintainability of safety-critical railway assets. In addition to that, the challenges that are currently slowing the process of integration of AI and DL in railways were considered, especially concentrating on the certifiability of AI. Some guidelines were proposed to gradually introduce AI in Autonomous Train driving, on the one hand, and support the proper development of DL systems for railway safety and maintenance applications, on the other hand. Then, experiments to showcase how DL could be applied to improve or support current practices in these directions were discussed. The hope is that the obtained results could inspire and support future research on the topics of vision-based Obstacle Detection and smart maintenance of safety-critical railway assets.

Bibliography

- [1] A. Annoni, P. Benczur, P. Bertoldi, B. Delipetrev, G. De Prato, C. Feijoo, E. Fernandez Macias, E. Gomez, M. Iglesias, H. Junklewitz, M. Lopez Cobo, B. Mertens, S. Nascimento, S. Nativi, A. Polvora, I. Sanchez, S. Tolan, I. Tuomi, and L. Vesnic Alujevic. Artificial Intelligence - A European Perspective. Technical report, Craglia, M. editor(s), EUR 29425 EN, Publications Office of the European Union, Luxembourg, 2018.
- [2] A. Antoniou, A. Storkey, and H. Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.
- [3] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1505.07293*, 2015.
- [4] D. Bank, N. Koenigstein, and R. Giryes. *Autoencoders*, pages 353–374. Springer International Publishing, Cham, 2023.
- [5] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.
- [6] M.A. Benzaghta, A. Elwalda, M.M. Mousa, I. Erkan, and M. Rahman. Swot analysis applications: An integrative literature review. *J. of Glob. Bus. Insights*, 6(1):55–73, 2021.
- [7] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2019.

- [8] N. Bešinović, L. De Donato, F. Flammini, R.M.P. Goverde, Z. Lin, R. Liu, S. Marrone, R. Nardone, T. Tang, and V. Vittorini. Artificial intelligence in railway transport: Taxonomy, regulations, and applications. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14011–14024, 2022.
 - [9] C.M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.
 - [10] M. Borg, J. Bronson, L. Christensson, F. Olsson, O. Lennartsson, E. Sonnsjö, H. Ebabi, and M. Karsberg. Exploring the assessment list for trustworthy ai in the context of advanced driver-assistance systems. In *2021 IEEE/ACM 2nd International Workshop on Ethics in Software Engineering Research and Practice (SEthics)*, pages 5–12. IEEE, 2021.
 - [11] A. Boussik, W. Ben-Messaoud, S. Niar, and A. Taleb-Ahmed. Railway obstacle detection using unsupervised learning: An exploratory study. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 660–667, 2021.
 - [12] M. Brucker, A. Cramariuc, C. von Einem, R. Siegwart, and C. Cadena. Local and global information in obstacle detection on railway tracks. *arXiv preprint arXiv:2307.15478*, 2023.
 - [13] Y. Cao, H. Pan, H. Wang, X. Xu, Y. Li, Z. Tian, and X. Zhao. Small object detection algorithm for railway scene. In *2022 7th International Conference on Image, Vision and Computing (ICIVC)*, pages 100–105, 2022.
 - [14] R. Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997.
 - [15] E. Castillo, J.M. Gutierrez, and A.S. Hadi. *Expert systems and probabilistic network models*. Springer Science & Business Media, 2012.
 - [16] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker. Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems*, 30, 2017.
 - [17] P. Chen, Y. Wu, Y. Qin, H. Yang, and Y. Huang. Rail fastener defect inspection based on uav images: a comparative study. In *Proceedings of the 4th International Conference on Electrical and Information Technologies for Rail Transportation (EITRT) 2019: Rail Transportation Information Processing and Operational Management Technologies*, pages 685–694. Springer, 2020.
 - [18] S.-X. Chen, L. Zhou, Y.-Q. Ni, and X.-Z. Liu. An acoustic-homologous transfer learning approach for acoustic emission-based rail condition evaluation. *Structural Health Monitoring*, 20(4):2161–2181, 2021.
-

-
- [19] Z. Chen, W. Niu, C. Wu, L. Zhang, and Y. Wang. Near real-time situation awareness and anomaly detection for complex railway environment. In *2021 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, pages 1–8, 2021.
 - [20] A. Chernov, M. Butakova, A. Guda, and P. Shevchuk. Development of intelligent obstacle detection system on railway tracks for yard locomotives using cnn. In S. Bernardi, V. Vittorini, F. Flammini, R. Nardone, S. Marrone, R. Adler, D. Schneider, P. Schlei, N. Nostro, R. Lovenstein Olsen, A. Di Salle, and P. Masci, editors, *Dependable Computing - EDCC 2020 Workshops*, pages 33–43, Cham, 2020. Springer International Publishing.
 - [21] A. Dagvasumberel, B. Myagmardulam, B. Myagmar, B. Luvsankhuu, and T. Nakayama. Railroad near-miss occurrence detection and risk estimation system with data from camera using deep learning. In *2021 5th International Conference on Imaging, Signal Processing and Communications (ICISPC)*, pages 83–87, 2021.
 - [22] N. Davari, B. Veloso, G. de Assis Costa, R.M. Pereira, R. P. Ribeiro, and J. Gama. A survey on data-driven predictive maintenance for the railway industry. *Sensors*, 21(17), 2021.
 - [23] L. De Donato, R. Dirnfeld, A. Somma, A. De Benedictis, F. Flammini, S. Marrone, M. Saman Azari, and V. Vittorini. Towards AI-assisted digital twins for smart railways: preliminary guideline and reference architecture. *Journal of Reliable Intelligent Environments*, pages 1–15, 2023.
 - [24] L. De Donato, F. Flammini, S. Marrone, C. Mazzariello, R. Nardone, C. Sansone, and V. Vittorini. A survey on audio-video based defect detection through deep learning in railway maintenance. *IEEE Access*, 10:65376–65400, 2022.
 - [25] L. De Donato, S. Marrone, F. Flammini, C. Sansone, V. Vittorini, R. Nardone, C. Mazzariello, and F. Bernaudin. Intelligent detection of warning bells at level crossings through deep transfer learning for smarter railway maintenance. *Engineering Applications of Artificial Intelligence*, 123:106405, 2023.
 - [26] L. De Donato, R. Tang, N. Beinovi, F. Flammini, R.M.P. Goverde, Z. Lin, R. Liu, S. Marrone, E. Napoletano, R. Nardone, S. Santini, and V. Vittorini. Artificial intelligence in railways: current applications, challenges, and ongoing research. *Handbook on Artificial Intelligence and Transport*, page 249, 2023.
-

-
- [27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
 - [28] Y. Deng, T. Zhang, G. Lou, X. Zheng, J. Jin, and Q.-L. Han. Deep learning-based autonomous driving systems: A survey of attacks and defenses. *IEEE Transactions on Industrial Informatics*, 17(12):7897–7912, 2021.
 - [29] C. Di Meo, M. Di Vaio, F. Flammini, R. Nardone, S. Santini, and V. Vittorini. ERTMS/ETCS virtual coupling: Proof of concept and numerical analysis. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2545–2556, 2019.
 - [30] G. D’Amico, M. Marinoni, F. Nesti, G. Rossolini, G. Buttazzo, S. Sabina, and G. Lauro. Trainsim: A railway simulation framework for lidar and camera dataset generation. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–0, 2023.
 - [31] EASA. EASA Concept Paper: First usable guidance for Level 1 machine learning applications. Technical report, 2021.
 - [32] EASA. EASA Concept Paper: First usable guidance for Level 1&2 machine learning applications. Technical report, 2021.
 - [33] EASA and Daedalean. Concepts of Design Assurance for Neural Networks (CoDANN) II. Technical report, 2021.
 - [34] European Commission. Communication from the commission to the European parliament, the European Council, the Council, the European economic and social committee and the committee of the regions. *Artificial Intelligence for Europe. Brussels, Belgium*, 2018.
 - [35] European Commission. Proposal for a Regulation of the European Parliament and of the Council laying down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and amending certain Union Legislative Acts. Brussels, Belgium, 2021.
 - [36] European Commission, Joint Research Centre, D. Fernández Llorca, and E. Gómez. *Trustworthy autonomous vehicles – Assessment criteria for trustworthy AI in the autonomous driving domain*. Publications Office of the European Union, 2021.
 - [37] European Commission and Directorate-General for Communications Networks, Content and Technology. *The Assessment List for Trustworthy Artificial Intelligence (ALTAI) for self assessment*. Publications Office, 2020.
-

-
- [38] European Commission, Directorate-General for Communications Networks, Content and Technology. *Ethics guidelines for trustworthy AI*. Publications Office, 2019.
 - [39] European Union Agency for Railways. Introduction to ETCS braking curves, 2020.
 - [40] European Union Agency for Railways. Safety overview 2023, 2023.
 - [41] J. Farooq. Object detection and identification using surf and bow model. In *2016 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube)*, pages 318–323, 2016.
 - [42] M.A.B. Fayyaz, A.C. Alexoulis-Chrysovergis, M.J. Southgate, and C. Johnson. A review of the technological developments for interlocking at level crossing. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, page 0954409720941726, 2020.
 - [43] J. Feng, X. Liang, and Y. Wu. Data fusion-based obstacle detection for rail transit scenarios. In *2022 World General Artificial Intelligence Congress*. SAE International, jun 2022.
 - [44] F. Flammini, L. De Donato, A. Fantechi, and V. Vittorini. A Vision of Intelligent Train Control. In S. Collart-Dutilleul, A. E. Haxthausen, and T. Lecomte, editors, *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification*, pages 192–208, Cham, 2022. Springer International Publishing.
 - [45] M. Franke, V. Gopinath, D. Ristić-Durrant, and K. Michels. Object-level data augmentation for deep learning-based obstacle detection in railways. *Applied Sciences*, 12(20), 2022.
 - [46] M. Franke, C. Reddy, D. Ristić-Durrant, J. Jayawardana, K. Michels, M. Banić, and M. Simonović. Towards holistic autonomous obstacle detection in railways by complementing of on-board vision with uav-based object localization. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7012–7019, 2022.
 - [47] A. Galli, S. Marrone, V. Moscato, and C. Sansone. Reliability of explainable artificial intelligence in adversarial perturbation scenarios. In *International Conference on Pattern Recognition*, pages 243–256. Springer, 2021.
 - [48] R. Gasparini, A. D’Eusano, G. Borghi, S. Pini, G. Scaglione, S. Calderara, E. Fedeli, and R. Cucchiara. Anomaly detection, localization and classification for railway inspection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3419–3426, 2021.
-

-
- [49] J.F. Gemmeke, D.P.W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R.C. Moore, M. Plakal, and M. Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Int. Conf. on Acoustics, Speech and Signal*, pages 776–780. IEEE, 2017.
 - [50] M. Gleirscher, A.E. Haxthausen, and J. Peleska. Probabilistic risk assessment of an obstacle detection system for GoA 4 freight trains. 2023.
 - [51] R. Goel, A. Sharma, and R. Kapoor. An efficient object and railway track recognition in thermal images using deep learning. In N. Marriwala, C.C. Tripathi, S. Jain, and S. Mathapathi, editors, *Emergent Converging Technologies and Biomedical Systems*, pages 241–253, Singapore, 2022. Springer Singapore.
 - [52] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
 - [53] J. Gou, B. Yu, S.J. Maybank, and D. Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.
 - [54] D. Grzechca, P. Rybka, and R. Pawełczyk. Level crossing barrier machine faults and anomaly detection with the use of motor current waveform analysis. *Energies*, 14(11), 2021.
 - [55] L. Guan, L. Jia, Z. Xie, and C. Yin. A lightweight framework for obstacle detection in the railway image based on fast region proposal and improved yolo-tiny network. *IEEE Transactions on Instrumentation and Measurement*, 71:1–16, 2022.
 - [56] A. Guesmi, M.A. Hanif, B. Ouni, and M. Shafique. Physical adversarial attacks for camera-based smart systems: Current trends, categorization, applications, research challenges, and future outlook. *IEEE Access*, 11:109617–109668, 2023.
 - [57] Q. Guo, L. Liu, W. Xu, Y. Gong, X. Zhang, and W. Jing. An improved faster r-cnn for high-speed railway dropper detection. *IEEE Access*, 8:105622–105633, 2020.
 - [58] M.A. Haseeb, J. Guan, D. Ristić-Durrant, and A. Gräser. DisNet: a novel method for distance estimation from monocular camera. *10th Planning, Perception and Navigation for Intelligent Vehicles (PPNIV18), IROS*, 10, 2018.
 - [59] M.A. Haseeb, D. Ristić-Durrant, A. Gräser, M. Banić, and D. Stamenković. Multi-DisNet: Machine learning-based object distance estimation from multiple cameras. In D. Tzovaras, D. Giakoumis, M. Vincze, and A. Argyros,
-

- editors, *Computer Vision Systems*, pages 457–469, Cham, 2019. Springer International Publishing.
- [60] D. He, Y. Qiu, J. Miao, Z. Zou, K. Li, C. Ren, and G. Shen. Improved Mask R-CNN for obstacle detection of rail transit. *Measurement*, 190:110728, 2022.
- [61] D. He, Z. Zou, Y. Chen, B. Liu, and J. Miao. Rail transit obstacle detection based on improved cnn. *IEEE Transactions on Instrumentation and Measurement*, 70:1–14, 2021.
- [62] K. He, G. Gkioxari, P. Dollár, and R.B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.
- [63] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [64] S. Hershey, S. Chaudhuri, D.P.W. Ellis, J.F. Gemmeke, A. Jansen, R.C. Moore, M. Plakal, D. Platt, R.A. Saurous, B. Seybold, et al. CNN architectures for large-scale audio classification. In *Int. Conf. on Acoustics, Speech and Signal Processing*, pages 131–135. IEEE, 2017.
- [65] H.-W. Huang, Q.-T. Li, and D.-M. Zhang. Deep learning based image recognition for crack and leakage defects of metro shield tunnel. *Tunnelling and underground space technology*, 77:166–176, 2018.
- [66] M. Iman, H.R. Arabnia, and K. Rasheed. A review of deep transfer learning and recent advancements. *Technologies*, 11(2), 2023.
- [67] International Electronical Commission (IEC). IEC 62267:2009 - Railway applications - Automated urban guided transport (AUGT) - Safety requirements, 2009.
- [68] K. Jahan, J.P. Umesh, and M. Roth. Anomaly detection on the rail lines using semantic segmentation and self-supervised learning. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7, 2021.
- [69] G. Jing, X. Qin, H. Wang, and C. Deng. Developments, challenges, and perspectives of railway inspection robots. *Automation in construction*, 138:104242, 2022.
- [70] R. Kapoor, R. Goel, and A. Sharma. An intelligent railway surveillance framework based on recognition of object and railway track using deep learning. *Multimedia tools and applications*, 81(15):21083–21109, 2022.
-

-
- [71] S. Khobragade, A. Kinage, D. Shambharkar, and A. Gandhi. Real-time track and anomaly detection in complex railway environment. In *2022 1st International Conference on the Paradigm Shifts in Communication, Embedded Systems, Machine Learning and Signal Processing (PCEMS)*, pages 93–97, 2022.
 - [72] L. Khoudour, M. Ghazel, F. Boukour, M. Heddebaut, and E.-M. El-Koursi. Towards safer level crossings: existing recommendations, new applicable technologies and a proposed simulation model. *European transport research review*, 1(1):35–45, 2009.
 - [73] Y. Ko, Y. Lee, S. Azam, F. Munir, M. Jeon, and W. Pedrycz. Key points estimation and point instance segmentation approach for lane detection. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):8949–8958, 2022.
 - [74] P. Koopman and M. Wagner. Toward a framework for highly automated vehicle safety validation. Technical report, SAE Technical Paper, 2018.
 - [75] C. Larman. *Applying UML and patterns: an introduction to object oriented analysis and design and interative development*. Pearson Education India, 2012.
 - [76] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
 - [77] D. Li, Y. Wang, W.-J. Yan, and W.-X. Ren. Acoustic emission wave classification for rail crack monitoring based on synchrosqueezed wavelet transform and multi-branch convolutional neural network. *Structural Health Monitoring*, 20(4):1563–1582, 2021.
 - [78] S. Li, H. Zhao, and J. Ma. An edge computing-enabled train obstacle detection method based on YOLOv3. *Wireless Communications and Mobile Computing*, 2021:1–9, 2021.
 - [79] T. Li, A.K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.
 - [80] Y. Li, D. Wei, X. Wei, K. Wu, Y. Liang, and X. Shen. Defects detection of catenary suspension device based on image processing and cnn. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1756–1761. IEEE, 2019.
 - [81] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick. Microsoft COCO: Common objects in context. In D. Fleet,
-

- T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.
- [82] G. Litjens, T. Kooi, B.E. Bejnordi, A.A.A. Setio, F. Ciompi, M. Ghafoorian, J.A.W.M. van der Laak, B. van Ginneken, and C.I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.
- [83] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128:261–318, 2020.
- [84] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A.C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016.
- [85] Z. Liu, Y. Lyu, L. Wang, and Z. Han. Detection approach based on an improved faster rcnn for brace sleeve screws in high-speed railways. *IEEE Transactions on Instrumentation and Measurement*, 69(7):4395–4403, 2019.
- [86] T. Lu, X. Chen, and C. Geng. Nflow-based arbitrary subway obstacle detection method. In *2022 IEEE 7th International Conference on Intelligent Transportation Engineering (ICITE)*, pages 486–492, 2022.
- [87] T. Lu, C. Geng, and H. Dai. Subway obstacle detection system based on machine vision. In Z. Zhang, editor, *2021 6th International Conference on Intelligent Transportation Engineering (ICITE 2021)*, pages 1115–1122, Singapore, 2022. Springer Nature Singapore.
- [88] S.M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.
- [89] M. Macchi, M. Garetti, D. Centrone, L. Fumagalli, and G.P. Pavirani. Maintenance management of railway infrastructures based on reliability analysis. *Reliability Engineering & System Safety*, 104:71–83, 2012.
- [90] T.A. Madiega. Artificial intelligence act. *European Parliament: European Parliamentary Research Service*, 2021.
- [91] S. Manzoor, S.-H. Joo, and T.-Y. Kuc. Comparison of object recognition approaches using traditional machine vision and modern deep learning techniques for mobile robot. In *2019 19th International Conference on Control, Automation and Systems (ICCAS)*, pages 1316–1321, 2019.
- [92] S. Marrone, S. Olivieri, G. Piantadosi, and C. Sansone. Reproducibility of deep cnn for biomedical image processing across frameworks and architectures. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2019.
-

- [93] M. Martinez, C. Sitawarin, K. Finch, L. Meincke, A. Yablonski, and A. Kornhauser. Beyond grand theft auto v for training, testing and enhancing deep learning in self driving cars. *arXiv preprint arXiv:1712.01397*, 2017.
 - [94] N. Mazzino, X. Perez, U. Meuser, R. Santoro, M. Brennan, J. Schlaht, C. Chéron, H. Samson, L. Dauby, N. Furio, and C. Hernandez. RAIL 2050 VISION. Rail - The backbone of Europe’s mobility. *ERRAC, the European Rail Research Advisory Council*, 2017.
 - [95] D. Milburn and M. Erskine. Digital train control: Functional safety for AI based systems. In *Intl. Railway Safety Council Conference 2019*, Perth, Australia, 2019.
 - [96] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.
 - [97] D. Münch, A.-K. Grosselfinger, E. Krempel, M. Hebel, and M. Arens. Data anonymization for data protection on publicly recorded data. In *Computer Vision Systems: 12th International Conference, ICVS 2019, Thessaloniki, Greece, September 23–25, 2019, Proceedings 12*, pages 245–258. Springer, 2019.
 - [98] F. Nesti, G. Rossolini, S. Nair, A. Biondi, and G. Buttazzo. Evaluating the robustness of semantic segmentation for autonomous driving against real-world adversarial patch attacks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2280–2289, 2022.
 - [99] Network Rail. NR/L2/XNG/19608 ISSUE 8, 2022.
 - [100] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool. Towards end-to-end lane detection: an instance segmentation approach, 2018.
 - [101] N. O’Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh. Deep learning vs. traditional computer vision. In K. Arai and S. Kapoor, editors, *Advances in Computer Vision*, pages 128–144, Cham, 2020. Springer International Publishing.
 - [102] D. Omeiza, H. Webb, M. Jirotko, and L. Kunze. Explanations in autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10142–10162, 2022.
 - [103] D. Orive, N. Iriondo, A. Burgos, I. Saráchaga, M. L. Álvarez, and M. Marcos. Fault injection in digital twin as a means to test the response to process faults at virtual commissioning. In *2019 24th IEEE International Conference*
-

- on Emerging Technologies and Factory Automation (ETFA)*, pages 1230–1234, 2019.
- [104] S.J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
 - [105] M.J. Pappaterra, F. Flammini, V. Vittorini, and N. Bešinović. A systematic review of artificial intelligence public datasets for railway applications. *Infrastructures*, 6(10), 2021.
 - [106] M.I. Pavel, S.Y. Tan, and A. Abdullah. Vision-based autonomous vehicle systems based on deep learning: A systematic literature review. *Applied Sciences*, 12(14):6831, 2022.
 - [107] M. Pavlović, M. Banić, M. Simonović, V. Nikolić, et al. Ai powered obstacle distance estimation for onboard autonomous train operation. *Tehnički vjesnik*, 29(2):611–619, 2022.
 - [108] J. Peleska, A.E. Haxthausen, and T. Lecomte. Standardisation considerations for autonomous train control. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Practice*, pages 286–307, Cham, 2022. Springer Nature Switzerland.
 - [109] A. Przegalinska. State of the art and future of artificial in-telligence. 2019.
 - [110] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, 2019.
 - [111] RAILS. Definition of a reference taxonomy of AI in railways. *RAILS Project Deliverable D1.1, Shift2Rail*, 2020.
 - [112] RAILS. Application Areas. *RAILS Project Deliverable D1.3, Shift2Rail*, 2021.
 - [113] RAILS. Summary of existing relevant projects and state-of-the-art of AI application in railways. *RAILS Project Deliverable D1.2, Shift2Rail*, 2021.
 - [114] RAILS. Report on identification of migration strategies and roadmaps for AI integration in the rail sector. *RAILS Project Deliverable D5.3, Shift2Rail*, 2023.
 - [115] RAILS. WP2 Report on experimentation, analysis, and discussion of results. *RAILS Project Deliverable D2.3, Shift2Rail*, 2023.
 - [116] RAILS. WP2 Report on identification of future innovation needs and recommendations for improvements. *RAILS Project Deliverable D2.4, Shift2Rail*, 2023.
-

-
- [117] RAILS. WP3 Report on experimentation, analysis, and discussion of results. *RAILS Project Deliverable D3.3, Shift2Rail*, 2023.
 - [118] RAILS. WP3 Report on identification of future innovation needs and recommendations for improvements. *RAILS Project Deliverable D3.4, Shift2Rail*, 2023.
 - [119] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
 - [120] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
 - [121] K. Ren, T. Zheng, Z. Qin, and X. Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.
 - [122] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
 - [123] M.T. Ribeiro, S. Singh, and C. Guestrin. "Why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.
 - [124] P. Richard, A. Boussif, and C. Paglia. Rule-based and managed safety: a challenge for railway autonomous driving systems. In *31th European Safety and Reliability Conference*, 2021.
 - [125] D. Ristić-Durrant, M. Franke, and K. Michels. A Review of Vision-Based On-Board Obstacle Detection and Distance Estimation in Railways. *Sensors*, 21(10), 2021.
 - [126] D. Ristić-Durrant, M.A. Haseeb, M. Franke, M. Banić, M. Simonović, and D. Stamenković. Artificial intelligence for obstacle detection in railways: Project SMART and beyond. In S. Bernardi, V. Vittorini, F. Flammini, R. Nardone, S. Marrone, R. Adler, D. Schneider, P. Schleiß, N. Nostro, R. Løvenstein Olsen, A. Di Salle, and P. Masci, editors, *Dependable Computing - EDCC 2020 Workshops*, pages 44–55, Cham, 2020. Springer International Publishing.
 - [127] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
-

-
- [128] SAFER-LC. Needs and requirements for improving level crossing safety. *SAFER-LC Project Deliverable D1.3*, 2018.
 - [129] J. Salamon and J.P. Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal processing letters*, 24(3):279–283, 2017.
 - [130] B. Simoni. ATO over ETCS, 2022. Accessed on 28 August 2023.
 - [131] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
 - [132] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
 - [133] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
 - [134] R. Tagiew, M. Köppel, K. Schwalbe, P. Denzler, P. Neumaier, T. Klockau, M. Boekhoff, P. Klasek, and R. Tilly. OSDaR23: Open sensor data for rail 2023. *arXiv preprint arXiv:2305.03001*, 2023.
 - [135] R. Tagiew, D. Leinhos, H. von der Haar, C. Klotz, D. Sprute, J. Ziehn, A. Schmelter, S. Witte, and P. Klasek. Sensor system for development of perception systems for ato. *Discover Artificial Intelligence*, 3(1):22, 2023.
 - [136] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 270–279, Cham, 2018. Springer International Publishing.
 - [137] P. Tan, X. Li, Z. Wu, J. Ding, J. Ma, Y. Chen, Y. Fang, and Y. Ning. Multialgorithm fusion image processing for high speed railway dropper failure–defect detection. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(7):4466–4478, 2019.
 - [138] R. Tang, L. De Donato, N. Besšinović, F. Flammini, R.M.P. Goverde, Z. Lin, R. Liu, T. Tang, V. Vittorini, and Z. Wang. A literature review of Artificial Intelligence applications in railway systems. *Transportation Research Part C: Emerging Technologies*, 140:103679, 2022.
 - [139] The European Rail Research Advisory Council. RAIL 2030 - Research and innovation priorities, 2019.
-

-
- [140] The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems. Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems, First Edition. 2019.
 - [141] A. M. Turing. Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer, 2009.
 - [142] Underwriters Laboratories Inc. ANSI/UL 4600-2020 standard for evaluation of autonomous products - first edition. Underwriters Laboratories Inc., 333 Pingsten Road, Northbrook, Illinois 60062-2096, 847.272.8800, 2020.
 - [143] E. Uygun and S. Terzi. Acoustic monitoring of railway defects using deep learning with audio to spectrogram conversion. *Journal of Vibration Engineering & Technologies*, pages 1–10, 2023.
 - [144] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.
 - [145] J. Wang, L. Luo, W. Ye, and S. Zhu. A defect-detection method of split pins in the catenary fastening devices of high-speed railway based on deep learning. *IEEE Transactions on Instrumentation and Measurement*, 69(12):9517–9525, 2020.
 - [146] L. Wang, D. Zhang, J. Guo, and Y. Han. Image anomaly detection using normal data only by latent space resampling. *Applied Sciences*, 10(23), 2020.
 - [147] Y. Wang, L. Wang, Y.H. Hu, and J. Qiu. RailNet: A segmentation network for railroad detection. *IEEE Access*, 7:143772–143779, 2019.
 - [148] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
 - [149] Z. Wang, X. Wu, G. Yu, and M. Li. Efficient rail area detection using convolutional neural network. *IEEE Access*, 6:77656–77664, 2018.
 - [150] M. Wedberg. Detecting rails in images from a train-mounted thermal camera using a convolutional neural network, 2017.
 - [151] X. Wei, D. Wei, D. Suo, L. Jia, and Y. Li. Multi-target defect identification for railway track line based on image processing and improved yolov3 model. *IEEE Access*, 8:61973–61988, 2020.
 - [152] I.H. Witten, E. Frank, M.A. Hall, and C.J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2016.
-

-
- [153] X2RAIL-4. GoA3/4 Specification. *X2RAIL-4 Project Deliverable D3.2, Shift2Rail*, 2022.
 - [154] X. Xu, C.-L. Li, and Z. Xu. Integrated train timetabling and locomotive assignment. *Transportation Research Part B: Methodological*, 117:573–593, 2018.
 - [155] Y. Xu, C. Gao, L. Yuan, S. Tang, and G. Wei. Real-time obstacle detection over rails using deep convolutional neural network. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1007–1012, 2019.
 - [156] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.
 - [157] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen. Image data augmentation for deep learning: A survey. *arXiv preprint arXiv:2204.08610*, 2022.
 - [158] Y. Yang, Y. Hu, L. Chen, x. Liu, N. Qin, and Z. Liu. Defect detection of axle box cover device fixing bolts in metro based on convolutional neural network. In *2020 39th Chinese Control Conference (CCC)*, pages 7504–7509. IEEE, 2020.
 - [159] T. Ye, C. Ren, X. Zhang, G. Zhai, and R. Wang. Application of lightweight railway transit object detector. *IEEE Transactions on Industrial Electronics*, 68(10):10269–10280, 2021.
 - [160] T. Ye, X. Zhang, Y. Zhang, and J. Liu. Railway traffic object detection using differential feature fusion convolution neural network. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1375–1387, 2021.
 - [161] T. Ye, Z. Zhang, X. Zhang, and F. Zhou. Autonomous railway traffic object detection using feature-enhanced single-shot detector. *IEEE Access*, 8:145182–145193, 2020.
 - [162] T. Ye, Z. Zhao, S. Wang, F. Zhou, and X. Gao. A stable lightweight and adaptive feature enhanced convolution neural network for efficient railway transit object detection. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):17952–17965, 2022.
 - [163] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. BiSeNet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018.
-

- [164] M. Yu, P. Yang, and S. Wei. Railway obstacle detection algorithm using neural network. *AIP Conference Proceedings*, 1967(1):040017, 2018.
 - [165] O. Zendel, M. Murschitz, M. Zeilinger, D. Steininger, S. Abbasi, and C. Beleznaï. RailSem19: A dataset for semantic rail scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.
 - [166] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.
 - [167] T. Zhu, Z. Zhang, C. Wu, Z. Zhang, T. He, H. Zhang, R. Manmatha, M. Li, and A. Smola. Improving semantic segmentation via self-training. *arXiv preprint arXiv:2004.14960*, 2020.
 - [168] M. Ziegler, V. Mhasawade, M. Köppel, P. Neumaier, and V. Eiselein. A comprehensive framework for evaluating vision-based on-board rail track detection. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–8, 2023.
-

Author's publications

1. N. Bešinović, L. De Donato, F. Flammini, R.M.P Goverde, Z. Lin, R. Liu, S. Marrone, R. Nardone, T. Tang, and V. Vittorini. Artificial Intelligence in Railway Transport: Taxonomy, Regulations and Applications. *IEEE Transactions on Intelligent Transportation Systems*, 23(9), 14011-14024, 2022. DOI: 10.1109/TITS.2021.3131637.
2. R. Tang, L. De Donato, N. Bešinović, F. Flammini, R.M.P. Goverde, Z. Lin, R. Liu, T. Tang, V. Vittorini, and Z. Wang. A Literature Review of Artificial Intelligence Applications in Railway Systems. *Transportation Research Part C: Emerging Technologies*, 140, 103679, 2022. DOI: 10.1016/j.trc.2022.103679.
3. L. De Donato, F. Flammini, S. Marrone, C. Mazzariello, R. Nardone, C. Sansone, and V. Vittorini. A Survey of Deep Learning Applications to Railway Maintenance by Audio-Video Analytics. *IEEE Access*, 10, 65376 - 65400, 2022. DOI: 10.1109/ACCESS.2022.3183102.
4. L. De Donato, S. Marrone, F. Flammini, C. Sansone, V. Vittorini, R. Nardone, C. Mazzariello, and F. Bernaudin. Intelligent Detection of Warning Bells at Level Crossings through Deep Transfer Learning for Smarter Railway Maintenance. *Engineering Applications of Artificial Intelligence*, 123, 106405, 2023. DOI: 10.1016/j.engappai.2023.106405.
5. L. De Donato, R. Dirnfeld, A. Somma, A. De Benedictis, F. Flammini, S. Marrone, M. Saman Azari, and V. Vittorini. Towards AI-assisted digital twins for smart railways: preliminary guideline and reference architecture. *Journal of Reliable Intelligent Environments*, 9, 303–317, 2023. DOI: 10.1007/s40860-023-00208-6.
6. R. Dirnfeld Túrocy, L. De Donato, A. Somma, M. Saman Azari, S. Marrone, F. Flammini, and V. Vittorini. Integrating AI and DTs: Challenges and

- Opportunities in Railway Maintenance Application and Beyond. SIMULATION (Accepted for publication, October 2023).
7. F. Flammini, L. De Donato, A. Fantechi, and V. Vittorini. A Vision of Intelligent Train Control. In: S. Collart-Dutilleul, et al. (eds), Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification. RSSRail 2022. Lecture Notes in Computer Science, vol. 13294, pp. 192-208, 2023. Springer, Cham. DOI: 10.1007/978-3-031-05814-1_14
 8. L. De Donato, F. Flammini, S. Marrone, R. Nardone, and V. Vittorini. Trustworthy AI for safe autonomy of smart railways: directions and lessons learnt from other sectors. World Congress on Railway Research (WCRR 2022). Birmingham, United Kingdom, June 2022.
 9. R. Dirnfeld, L. De Donato, F. Flammini, M. Saman Azari, and V. Vittorini. Railway Digital Twins and Artificial Intelligence: Challenges and Design Guidelines. In: S. Marrone, et al. (eds), Dependable Computing – EDCC 2022 Workshops. EDCC 2022. Communications in Computer and Information Science, vol. 1656, p. 102-113, 2022. Springer, Cham. DOI: 10.1007/978-3-031-16245-9_8
 10. L. De Donato, A. Ferraro, A. Galli, M. Gravina, S. Marrone, V. Moscato, G. Sperli, V. Vittorini, and C. Sansone. Advanced AI-based approaches in Industry 4.0 of the University of Naples Federico II node of the CINI-AIIS Lab. Proceedings of the Ital-IA 2023 Thematic Workshops co-located with the 3rd CINI National Lab AIIS Conference on Artificial Intelligence (Ital IA 2023), CEUR Workshop Proceedings, vol. 3486, p. 1-6, 2023.
 11. L. De Donato, F. Flammini, S. Marrone, R. Nardone, and V. Vittorini. Recommendations and Roadmaps Towards Intelligent Railways. 10th Transport Research Arena (TRA 2024). Dublin, Ireland, Apr. 2024 (Paper Submitted).
 12. L. De Donato, R. Tang, N. Bešinović, F. Flammini, R.M.P. Goverde, Z. Lin, R. Liu, S. Marrone, E. Napoletano, R. Nardone, S. Santini, and V. Vittorini. Artificial intelligence in railways: current applications, challenges, and ongoing research. In H. Dia (Ed.), Handbook on Artificial Intelligence and Transport. Edward Elgar Publishing, October 2023.
-